UNIVERSIDAD AUTÓNOMA METROPOLITANA

Casa abierta al tiempo

Maestría en Ciencias de la Computación

Título de la tesis

"Detección y clasificación de objetos dentro de un salón de clases empleando técnicas de procesamiento digital de imágenes"

Presenta:

- *Elías García Santillán*

Asesor:

- *Dr. Carlos Avilés Cruz*

Mayo de 2008

AGRADECIMIENTOS

A dios

Por darme la oportunidad de seguir adelante.

A mi familia

Por su comprensión y apoyo.

A mis profesores de la UAM

Por su dedicación y todas las atenciones recibidas.

RESUMEN

El presente trabajo describe las técnicas empleadas para realizar un sistema de reconocimiento automático de objetos. Inicialmente, este sistema de reconocimiento se planteó para ayudar a personas invidentes en el reconocimiento de objetos en su entorno; pero debido a la problemática existente en el diseño de algoritmos generales para el reconocimiento, se propuso la implementación del proyecto dentro de un salón de clases, limitándose a reconocer 10 objetos pequeños. En el primer capítulo realiza una pequeña introducción sobre la visión artificial y muestra la importancia de utilizar una máquina para la detección de objetos en la industria, en la medicina y en muchos campos del conocimiento; en este capítulo también se realiza una descripción general de nuestro sistema a elaborar y destacando la forma "*piramidal*" de reconocimiento, teniendo como primer criterio de clasificación el color del objeto, como segundo criterio la forma del objeto y por último, el reconocimiento por su textura. En el segundo capítulo se muestran algunos trabajos de tesis de maestría y tesis doctórales que utilizan técnicas para el reconocimiento automático de objetivos y que son de gran importancia en el desarrollo de nuestro proyecto. A partir del tercer capítulo y hasta el séptimo, se describen y analizan cada una de las etapas del reconocimiento. En el octavo capítulo se analiza el *software* de edición de audio: "*CoolEdit*" y en el noveno capítulo se realiza un análisis de resultados obtenidos al implementar el proyecto.

ABSTRACT

The present work describes techniques employees to carry out a system of automatic objects recognition. Initially, recognition system was planned to help blind people in the objects recognition around their environment; but due to the existent problem in the design of general algorithms for the recognition, A proposal was given for its implementation within a classroom, being limited only to the recognition of 10 small objects. In the first chapter carries out a small introduction on artificial vision and it shows the importance of using a machine for the detection of objects in the industry, in the medicine and in many fields of the knowledge; in this chapter is also carried out a general description of our system to elaborate and highlighting the pyramidal form of recognition, having as first classification approach the color of the object, as second approach the form of the object and lastly, the recognition for their texture. In the second chapter some works of master thesis and thesis doctoral are shown that use techniques for the automatic recognition of objectives and that they are of great importance in the development of our project. Starting from the third chapter and up to the seventh, they are described and they analyze each one of the recognition stages. In the eighth chapter the software of audio edition is analyzed: "CoolEdit" and in the ninth chapter is carried out an analysis of results obtained when implementing the project.

OBJETIVOS

La elaboración de esta tesis tuvo como fin el desarrollar un sistema de reconocimiento de objetos a través de la computadora por medio del procesamiento digital de imágenes. Para su elaboración fue necesario cumplir lo siguiente:

- Poner en práctica todos los conocimientos adquiridos en las materias de la maestría en ciencias de la computación.
- Realizar una amplia investigación en libros, tesis y artículos internacionales sobre las diferentes técnicas de procesamiento digital de imágenes.
- Diseñar un sistema de captación de imágenes por medio de una WebCam.
- Buscar y optimizar un algoritmo de mejoramiento de la imagen.
- Implementar la segmentación de objetos de forma automática (método de Otsu).
- Aplicar el método de reconocimiento piramidal en el reconocimiento de objetos para mejorar el grado de certeza del objeto.
- Realizar un análisis de resultados con estadísticas y gráficas para cada uno de los descriptores utilizados y también en forma global.
- Utilizar un láser y algunas fórmulas trigonométricas para el cálculo de la distancia del objeto.
- Buscar un programa de edición de audio para la grabación y reproducción de archivos de audio.
- Elaborar en un lenguaje de programación (MatLab) cada una de las rutinas necesarias en la elaboración del sistema de reconocimiento.

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

Capítulo 1

Capítulo 2

Capítulo 3

Capítulo 4

Capítulo 5

Capítulo 6

Capítulo 7

Capítulo 8

Capítulo 9

ÍNDICE DE TABLAS

Capítulo 2

Capítulo 4

Capítulo 6

Capítulo 8

Capítulo 9

ÍNDICE DE PROGRAMAS EN MATLAB

Capítulo 1

INTRODUCCIÓN

1.1 Visión por computadora

La visión es uno de los mecanismos sensoriales de percepción más importantes que tiene el ser humano y la mayoría de los organismos biológicos; la utilizamos para desenvolvernos eficientemente dentro del ambiente que nos rodea y detectar los objetos que son de nuestro interés por medio de su forma, color, relieve, dimensiones, distancia a la que se encuentra, etcétera.

La visión por computadora es la capacidad de la máquina para ver el mundo que le rodea, para deducir la estructura y las propiedades del mundo tridimensional a partir de una o más imágenes bidimensionales. La implantación de habilidades en una máquina como la de detectar y determinar la identidad de los objetos no sólo liberan al hombre de tareas tediosas y peligrosas sino también permite la realización de algunas otras tareas imposibles de realizar para el ser humano.

El problema de visión por computadora es un problema abierto para el cuál no existe algún algoritmo eficaz que reconozca todo tipo de objetos en cualquier ambiente y en el tiempo en que nuestro sentido de la vista lo realiza.

1.2. Aplicaciones de visión artificial

Existen cientos de algoritmos para el reconocimiento automático de objetivos (*Automatic Target Recognition*) para algún caso en particular, por ejemplo: la industria automotriz la utiliza para el reconocimiento y ensamblado de piezas [11]; en otras empresas se usa para la detección de placas de autos [2], el reconocimiento de personas [23], seguimiento automático de objetos [4], recolección de frutos [5], conteo de bacterias [10], usos militares [22], juego de fútbol en competencias de robots [25], el reconocimiento automático de señales de tránsito [21], entre muchas otras aplicaciones (*ver figura1.2.1*).

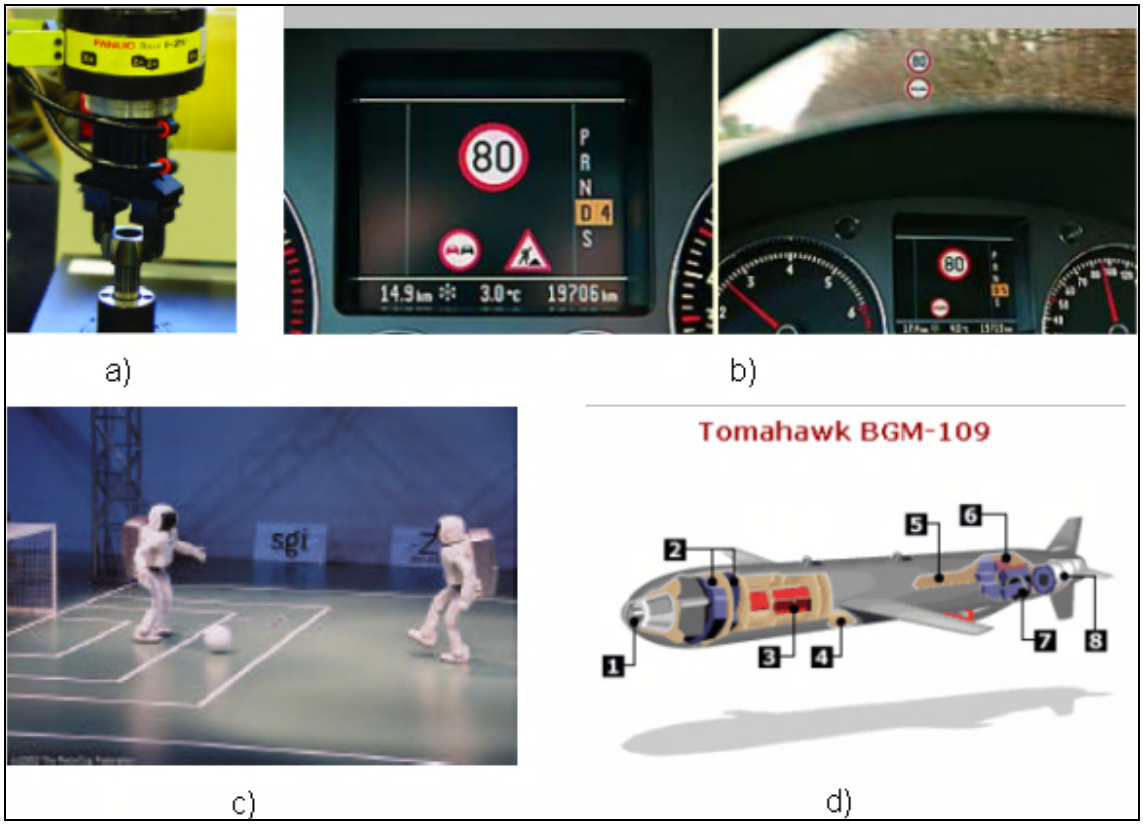


Figura 1.2.1 Aplicaciones de Visión Artificial. a) Ensamble de piezas por un robot, b) Reconocimiento automático de señales de tránsito. c) Juego de fútbol con humanoides. d) Usos militares.

1.3 Aplicaciones con tratamiento digital de imágenes

El procesamiento de imágenes también tiene muchas aplicaciones; por ejemplo, en la medicina se utiliza para el mejoramiento de imágenes obtenidas con fines de diagnóstico médico [35]. Alterando los valores de la luminosidad de los píxeles mediante transformaciones matemáticas en imágenes se logra detectar fallas de estructuras metálicas [9]. Otro ejemplo es el mejoramiento de imágenes aéreas para realizar exámenes de diagnóstico del terreno, localizar plantíos de droga, analizar recursos naturales, las fallas geológicas, etcétera. En las figuras 1.3.1 y 1.3.2 se muestran dos ejemplos sobre el tratamiento digital de imágenes.

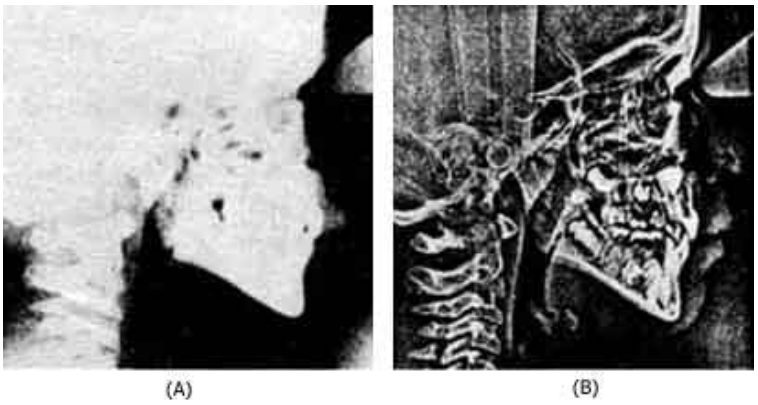


Figura 1.3.1. Cefalograma: A) Imagen original y B) imagen procesada.

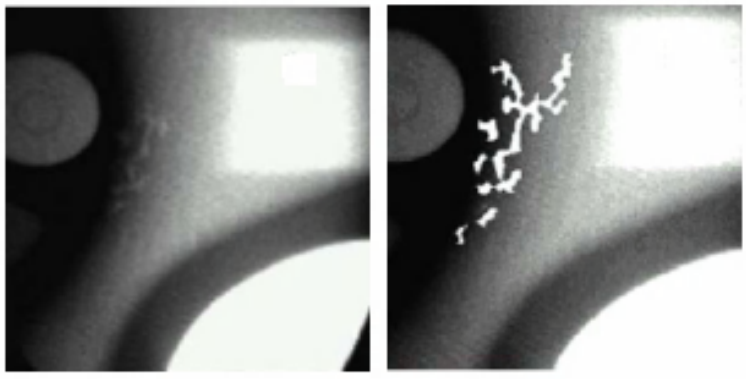


Figura 1.3.2. Fallas en estructuras: a) Imagen original, b) imagen procesada.

1.4 Etapas del reconocimiento automático de objetivos

El proceso para el reconocimiento automático de objetivos se inicia con la Adquisición de la imagen de una escena en tres dimensiones, a continuación esta imagen se procesa para mejorar la calidad y eliminar posibles imperfecciones; el siguiente paso es separar el objeto de interés del fondo de la imagen, seguida de la extracción de sus características que describen al objeto (*color, textura y geometría*), para finalmente, comparar estas características con las de otros objetos que se tienen en la base de conocimiento y así determinar el tipo de objeto (*figura 1.4.1*) [7].

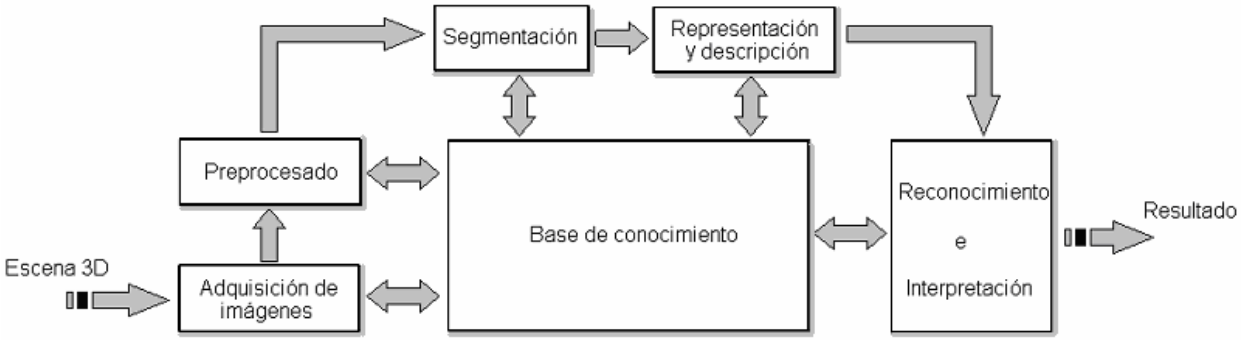


Figura 1.4.1: Etapas fundamentales en el reconocimiento automático de objetivos

En cada una de las etapas anteriores se requiere de un conocimiento previo, como puede ser: el tamaño del objeto, la distancia a la que se encuentra, las condiciones ambientales, la inclinación, el número de objetos a reconocer en la imagen, entre otros. Toda esta información es necesaria para la utilización de determinadas técnicas y el desarrollo de algoritmos adicionales para el reconocimiento del objeto.

Cada una de estas etapas se describirán con más detalle en los siguientes capítulos de esta tesis.

1.5 Descripción del proyecto

El sistema de visón de reconocimiento de objetos que se presenta en esta tesis inicialmente se planteó para personas invidentes; pero debido a la problemática existente en el diseño de algoritmos generales eficaces, y diversos factores que intervienen en su desarrollo y que son de gran importancia para la obtención de buenos resultados, nuestro sistema se implementa dentro de un salón de clases y se limita a reconocer 10 objetos pequeños.

Se utilizó una videocámara en la captación de la imagen y un láser para el cálculo de la distancia del objeto, a continuación se procesa esta imagen por medio de nuestro sistema de reconocimiento para su clasificación y finalmente se muestra en forma audible el nombre del objeto identificado y su distancia con respecto de la videocámara (*figura 1.5.1*).

Figura 1.5.1 Diagrama de bloques del sistema

El método de reconocimiento se implementa en forma piramidal (*figura 1.5.2*) teniendo como primer criterio el color del objeto, como segundo criterio la forma del objeto y por último el reconocimiento por su textura. Con esta técnica se pretende mejorar el porcentaje de certeza para el reconocimiento del objeto utilizando tres técnicas de reconocimiento. Si al implementar el reconocimiento por color tenemos una certeza mayor al 90%, no será necesario utilizar las otras dos técnicas; de lo contrario, se utiliza el reconocimiento por forma y si no fueron suficientes los primeros dos métodos para reconocer con certeza el objeto en cuestión, se utiliza el reconocimiento por su textura.

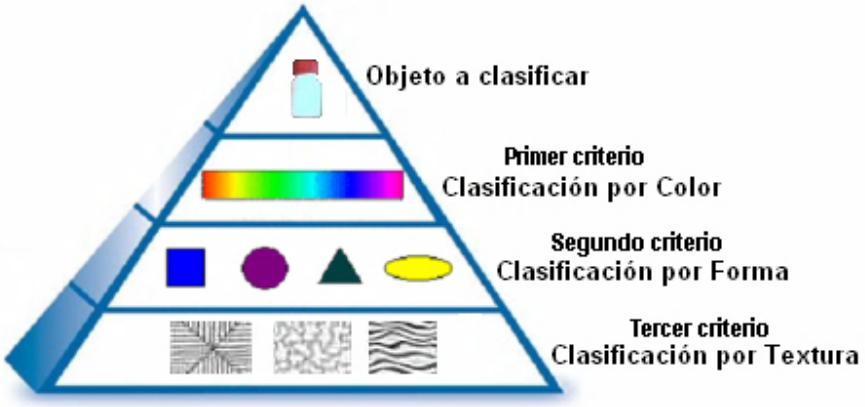


Figura 1.5.2: Método de reconocimiento piramidal

Los objetos a reconocer *(figura 1.5.3*) son pequeños con características muy variadas en cuanto al color, textura y forma.

Por ejemplo: el color del lápiz puede variar (*amarillo, blanco, morado, etc.*), puede tener goma, punta, estar un poco usado, entre otras características. Cada tipo de lápiz debe estar registrado en la base de conocimientos de nuestro sistema con sus características propias, y tal vez no se pueda reconocer directamente por el color debido a que el color del lápiz puede ser el mismo de otro objeto a reconocer, pero la textura puede variar o la forma alargada del lápiz marcará la diferencia. En síntesis, se requiere que un objeto contenga las mismas características de color, textura y forma para confundirse con otro objeto distinto.

El reconocimiento se realiza en un ambiente controlado como el salón de clases y con una buena iluminación ya que, como se verá en el capitulo 3, la iluminación juega un papel determinante en el éxito o fracaso del proyecto. Para ello fue necesario agregar luz incandescente (un foco de 100 wats) y apagar la luz fluorescente (luz blanca), cerrando la puerta y cubriendo las ventanas para que la luz externa no influya sobre la captación de la imagen.

Para evitar sombras y reflejos de luz, se utilizó un papel color negro. Además, el objeto a reconocer debe de estar en una posición en la cuál se le puedan extraer de la mejor manera sus características y no debe de haber otros objetos transpuestos.

Figura 1.5.3: Objetos a reconocer por nuestro sistema.

Capítulo 2

ESTADO DEL ARTE

En esta etapa se recopiló y analizó información que sirve como base para el desarrollo del proyecto que deseamos elaborar. Nos Ayuda a conocer la problemática a la que nos enfrentamos al implementar nuestro sistema y analizar las posibles soluciones que se plantean para este proyecto.

Los artículos presentados en este capítulo, principalmente las tesis de maestría y tesis doctórales son muy importantes ya que muestran un panorama muy amplio sobre las técnicas empleadas en el reconocimiento automático de objetivos (*RAO*).

También se analizaron otros artículos que no fueron de gran importancia para el desarrollo del proyecto ya que en algunos casos solo se menciona la técnica que emplearon para el reconocimiento, pero no se explica como se realizó ni que algoritmos o fórmulas emplearon; por lo tanto, no se mencionan en esta tesis.

2.1. Probando el desempeño de varias combinaciones de clasificadores y vectores de rasgos en la determinación de la identidad y el número de objetos circulando en una banda transportadora.

Karla Penélope Xicotencatl Aguilar. Tesis de maestría, 2001, Ed. Instituto Politécnico Nacional.

Resumen

Conocer el tipo y la cantidad de objetos circundando en una banda transportadora es muy importante; esto permitiría a un brazo manipulador, localizar y planear la trayectoria para tomar dichos objetos y realizar con ellos una tarea especificada.

Un objeto puede aparecer en la banda en posiciones variadas. Cuando dicho objeto es tomado por el brazo manipulador puede aparecer incluso más chico o más grande, dependiendo de la distancia y posición de dicho sensor al objeto. Otro punto importante para su reconocimiento es considerar las sombras y los cambios de iluminación.

Los tres clasificadores probados son:

- Distancia mínima.
- Distancia de Mahalanobis.
- Bayesiano.

Los descriptores usados son:

- Invariantes a traslaciones, rotaciones y cambios de escala de Hu
- Invariantes a afinidades propuestos por Flusser y Suk.

Un sistema de reconocimiento de patrones opera con todos los posibles objetos individuales que se van a reconocer. Estos objetos suelen denominarse patrones.

Existen varias maneras de reconocer un patrón. Una de ellas consiste en transformar dicho patrón en una tupla "*X*" cuyas componentes se denominan rasgos o características (*ver figura 2.1.1*). Cada tupla de rasgos se compara con un diccionario de tuplas preestablecidas, compuesto por las tuplas de rasgos de todos los objetos del universo de trabajo.

Figura 2.1.1 Objetos convertidos en tuplas de características

Se puede observar que los objetos individuales se convierten en tuplas "*X*" de rasgos antes de ser reconocidos que corresponden a las características del objeto. La notación usada para las tuplas de rasgos es de tipo columna:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} \qquad (2.1.1)$$

en donde, x1,x2, ... ,xp, pueden ser cualquier tipo de variable.

Una vez calculada la tupla "*X*" asociada a un objeto individual, su reconocimiento se basa en determinar su grado de semejanza con las tuplas previamente definidas. La selección del número y tipo de rasgos descriptores será tal que patrones individuales puedan ser discriminados o diferenciados. Es deseable que el valor de cada rasgo sea también invariante ante transformaciones.

El método para probar el desempeño de los clasificadores y descriptores seleccionados consta de dos etapas:

1) Etapa de obtención de funciones discriminantes: Este procedimiento describe los pasos de la etapa de entrenamiento para cada combinación clasificador-vector de rasgos. Dado una combinación clasificador-vector de rasgos, *Ci* y un conjunto de objetos a reconocer, *Oj, j = 1,...,N* realizar los siguientes pasos:

- Obtener *P* imágenes del objeto *Oj*
- Para cada imagen del objeto *Oj*. Aislar el objeto y calcular sus invariantes de *Hu* y de *Flusser*.
- Obtener los vectores medios y matrices de covarianza respectivos, según sea el caso.

2) Etapa de prueba: Dada una imagen f, conteniendo una o más instancias de uno o más objetos del universo de objetos usados durante la etapa de entrenamiento realizar los siguientes pasos:

1. Obtener la versión binaria b de la imagen f.
2. Aplicar cualquier algoritmo de etiquetado sobre b para obtener las regiones R_k $k=1,...,M$ de cada objeto.
3. Para cada R_k , obtener los rasgos característicos.
4. Aplicar la combinación C_i a cada R_k para obtener su clase correspondiente.
5. Obtener las estadísticas de desempeño para cada combinación clasificador-vector de rasgos.

El procedimiento anterior da como resultado los porcentajes de aciertos para cada combinación de clasificador-vector de rasgos.

Pruebas de desempeño

Los seis objetos usados para la experimentación se muestran en la figura 2.1.2., en ellos existe la presencia de sombras y brillos, que como es sabido influyen fuertemente en los procesos de entrenamiento.

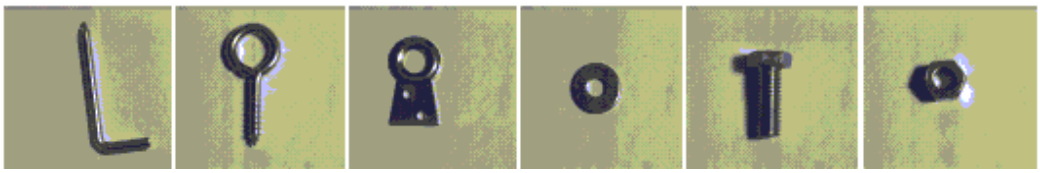


Figura 2.1.2. Objetos a reconocer.

En todos los casos, se probó el desempeño de las siguientes combinaciones clasificador-vector de rasgos:

1. Clasificador de distancia mínima: vector completo de Hu $[\phi_1, \phi_2, \phi_3, \phi_4, \phi_5, \phi_6, \phi_7]$
2. Clasificador de distancia mínima: vector reducido de Hu $[\phi_4, \phi_5, \phi_6, \phi_7]$
3. Clasificador de distancia mínima: vector completo de Flusser-Suk $[I_1, I_2, I_3, I_4, I_5, I_6]$
4. Clasificador de distancia mínima: vector reducido de Flusser-Suk $[I_2, I_4, I_5]$
5. Clasificador de distancia de Mahalanobis: vector completo de Hu $[\phi_1, \phi_2, \phi_3, \phi_4, \phi_5, \phi_6, \phi_7]$
6. Clasificador de distancia de Mahalanobis: vector reducido de Hu $[\phi_4, \phi_5, \phi_6, \phi_7]$
7. Clasificador de distancia de Mahalanobis: vector completo de Flusser-Suk $[I_1, I_2, I_3, I_4, I_5, I_6]$
8. Clasificador de distancia de Mahalanobis: vector reducido de Flusser-Suk $[I_2, I_4, I_5]$
9. Clasificador Bayesiano: vector completo de Hu $[\phi_1, \phi_2, \phi_3, \phi_4, \phi_5, \phi_6, \phi_7]$
10. Clasificador Bayesiano:: vector reducido de Hu $[\phi_4, \phi_5, \phi_6, \phi_7]$
11. Clasificador Bayesiano: vector completo de Flusser-Suk $[I_1, I_2, I_3, I_4, I_5, I_6]$
12. Clasificador Bayesiano: vector reducido de Flusser-Suk $[I_2, I_4, I_5]$

Se usaron vectores reducidos ya que se quería verificar si el desempeño de estos vectores con los respectivos clasificadores daba un mejor desempeño. Los vectores se obtuvieron al escoger aquellos invariantes con medias más separadas.

Desempeño de las combinaciones sin cambios de iluminación

Se tomaron 25 imágenes para cada una de las inclinaciones de 0, 22.5, 25 y 67.5 grados del sensor con respecto a la vertical a diversas distancias para obtener las deformaciones requeridas. El total de objetos reconocidos y porcentaje de reconocimiento de cada combinación se muestra en la tabla 2.1.1.

Podemos notar que el clasificador *Bayesiano–Flusser-Suk* da el mejor resultado. El segundo lugar lo ocupó la combinación *Mahalanobis-vector reducido de Flusser-Suk*.

Cambio de iluminación No.	Total de objetos reconocidos	Porcentaje de aciertos
1	316	52.6
2	323	53.8
3	359	59.8
4	342	57.0
5	455	75.8
6	418	69.6
7	462	77.0
8	469	78.1
9	463	77.1
10	451	75.1
11	477	79.5
12	449	74.8

Tabla 2.1.1 Desempeño de la combinación Bayesiano-Flusse-Suk.

Se probó el desempeño de la combinación *Bayesiano-Flusser-Zuk* ante cambios de iluminación dado que en el experimento anterior se obtuvieron mejores resultados. Se usaron 4 cambios de iluminación. Los resultados obtenidos se muestran en la tabla 2.1.2, obteniendo que el desempeño de la combinación sube y después baja al final del cuarto nivel de iluminación.

Cambio de iluminación No.	Total de objetos reconocidos	Porcentaje de aciertos
1	405	67.5
2	469	78.1
3	477	79.5
4	470	78.3

Tabla 2.1.2 Desempeño de la combinación Bayesiano-Flusse-Suk ante cambios de iluminación.

Se probó también el desempeño de la combinación *Bayesiano-Flusser-Suk* con objetos pintados de negro para eliminar los brillos, y como era de esperarse, el desempeño del clasificador fue del 91.16%.

Otra prueba que se realizó fue determinar la identidad y número de objetos circulando por una banda transportadora. Para ello el sistema toma una imagen del objeto dentro del campo visual del sensor y calcula su centroide en el tiempo t1 para calcular la velocidad de la banda, toma una siguiente imagen calculando de nuevo el centroide del mismo objeto en el tiempo t2 y con esta información se estima la velocidad "*v*" como:

$$v = \frac{x_2 - x_1}{t_2 - t_1} \qquad (2.1.2)$$

donde, x1 y x2 son las coordenadas de los centros del objeto.

Se probó únicamente el desempeño de la combinación *Bayesiano-Flusser-Suk*, obteniéndose un desempeño del 96.66%. Los errores en la determinación del número de objetos circulando sobre la banda se debieron a los tiempos de procesamiento, los cambios de velocidad de la banda y, las sombras y brillos en los objetos.

2.2. Localización de placas en vehículos automotores.

Arturo Legarda Sáenz, Tesis de maestría, 2001, Ed. Instituto Tecnológico de Chihuahua.

Resumen

El proceso implementado consta de varias etapas: adquisición de imágenes de vehículos, reprocesamiento a las imágenes capturadas, extracción y selección de regiones candidatas a contener la placa del vehículo y la localización de la placa.

Adquisición de las imágenes

Fueron adquiridas a través de una videocámara *Soni* en formato *BMP* a 16 colores con *640x480 píxeles* de resolución.

Preprocesamiento

A las imágenes capturadas con baja resolución se les aplica un operador de textura con el fin de localizar regiones. Este operador se basa en la medición de la energía de la textura presente en las superficies:

$$Tif : I(m,n) \rightarrow I_t(m,n) \qquad (2.2.1)$$

donde: $I(m,n)$ es una imagen y $I_t(m,n)$ es la imagen generada por el operador de textura. *Tif* se define como *avg(var(vecindad i))*, representando *avg* la función promedio y *var* es le varianza alrededor del pixel i de la imagen considerando cuatro direcciones, dentro de un vecindario vertical, horizontal, diagonal derecha y diagonal izquierda en una ventana de 3x3. El operador de textura genera regiones marcadas por bordes fuertes. A estos bordes se les aplica un suavizado con un filtro Gaussiano con desviación estándar de 0.8. para ampliar las líneas de píxeles que conectan regiones.

Extracción De Regiones

Se elimina información no necesaria mediante binarización con valor de umbral de 120. Se eliminan bordes que no generan una región cerrada y se aplica un operador morfológico de apertura, el cual está dado por:

$$Ro = (Ra \Theta Rb) \oplus Rb \qquad (2.2.2)$$

donde, *Ra* es la región a la cual se le aplica la apertura, *Rb* el elemento estructural en que se basa la apertura, en nuestro caso es un rectángulo de 3x3 píxeles, y *Ro* corresponde a la región resultante después de la apertura que es el resultado de una erosión (*operador* Θ) de un área *Ra* por un elemento estructural *Rb* seguido por una dilatación *(operador* $\oplus$) usando el mismo elemento estructural. A continuación se seleccionan las regiones que cumplen con área mínima=2000 píxeles y máxima de 300000 que es casi la totalidad de la imagen y se crea una nueva imagen con los píxeles de la imagen original que cumplen las siguientes condiciones:

$$r(s,t) = \frac{\sum_m \sum_n [\, I(m,n) - \hat{I}(m,n) \,][W(m-s,n-t) - w]}{\{ \sum_m \sum_n [\, I(m,n) - \hat{I}(m,n) \,]^2 + \sum_m \sum_n [W(m-s,n-t) - w]^2 \}^{1/2}} \qquad (2.2.3)$$

donde: *I(m,n)* es la imagen, *Î (m,n)* es el promedio de la imagen, *W(m,n)* es el patrón y w el promedio de *W(m,n)*. El patrón representa el objeto a identificar, en este caso la placa, El patrón (*template*), se generó con las imágenes del promedio de varias placas de automóviles. Para la obtención de las imágenes de placas se procedió a seleccionar y recortar de la imagen correspondiente a la placa. Como se muestra en las figuras 2.2.1,2.2.2 y 2.2.3.

Fig.2.2.1 Imagen original

Fig. 2.2.2 operador de textura.

Fig. 2.2.3 Patrón de la placa

Al realizar la búsqueda de la placa en la imagen se obtuvo un porcentaje de localización menor al 40% el cual no es aceptable.

Para mejorar el desempeño del sistema se modificó el operador de textura que mide la energía de la textura con lo que las áreas uniformes se minimizan, quedando los bordes resaltados. Esta imagen de textura se combinó con la imagen original usando un operador "*and*" lógico. Obteniéndose una imagen en la que los bordes aparecen realzados al tener valores de gris altos, mientras que las regiones uniformes sin bordes desaparecen (*ver figuras 2.2.4 a 2.2.8*).

Fig. 2.2.4 Regiones seleccionadas

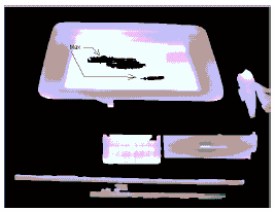


Fig.2.2.5 Función de correlación

Fig. 2.2.6 Operador and

Fig. 2.2.7 valores de correlación

Fig. 2.2.8 localización de la placa

Resultados

Los resultados obtenidos haciendo la modificación anterior tiene un 83% de localización correcta de la placa. Este desempeño aunque no es muy alto para un localizador es relativamente bueno comparado con sistemas que incluyen un conjunto de restricciones los cuales reportan rangos que van del 89% al 100% en la localización de la placa.

2.3. Nueva técnica para contar objetos en imágenes.

Rafael Sotelo Rangel, Tesis de maestría, 2001, Ed. Instituto Politécnico Nacional.

Resumen

El método del que habla el autor, se aplica a imágenes que contengan objetos semi-circulares, de contornos suaves, sin hoyos pero si puede contener traslapes entre objetos. El autor nos presenta varias definiciones, corolarios y teoremas para la mejor comprensión del tema (*ver figura 2.3.1*):

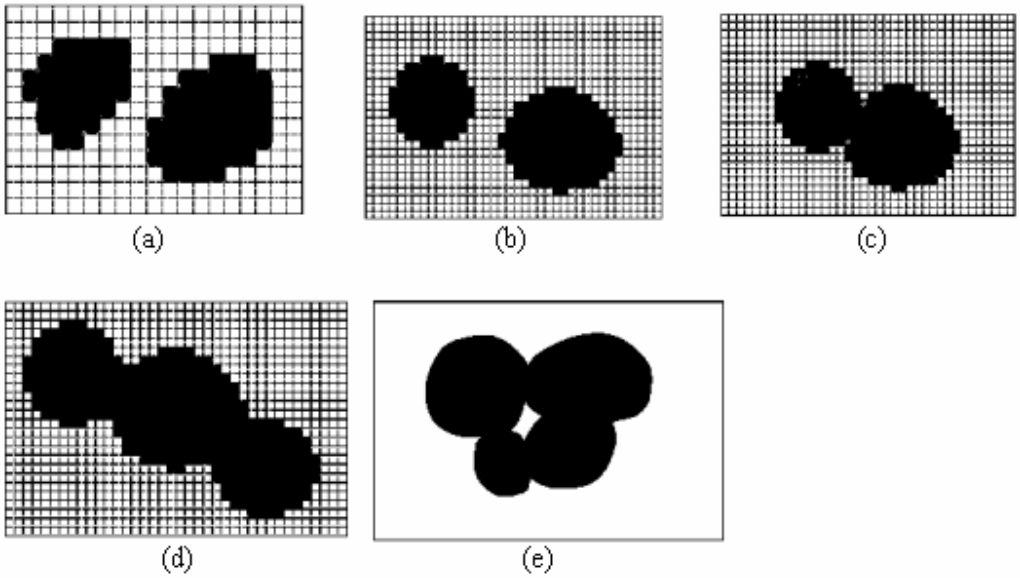


Figura 2.3.1: (a) Ejemplo de 2 blobs, (b) Ejemplo de 2 q-blob, (c) Puntos de concavidad, (d) Conglomerado, (e) Hoyo .

Un blob.- Es una región conectada por píxeles tal que entre cualesquiera de dos píxeles p1 y p2 de dicha región puede encontrar un camino conectando dichos píxeles.

Un q-blob.- Es un blob compacto semi-circular cuyo factor de circularidad asociado se encuentra muy cerca de la unidad.

Factor de circularidad, FC de una forma binaria es:

$$FC = 4\pi \frac{A}{P^2} \qquad (2.3.1)$$

donde, P es el perímetro del objeto y A su área.

Punto de concavidad singular.- es un punto de contorno resultado del traslape de dos q-blobs

Un conglomerado.- Es un conjunto de q-blobs traslapándose unos con otros.

Un hoyo.- Es una región tipo fondo resultado del traslape de varios q-blobs.

Lema 1. La manera de adicionar dos puntos de concavidad singular a un conglomerado consiste en traslapar un q-blob con dicho conglomerado sin generar un hoyo (ver figura 2.3.2 a).

Lema 2. La manera en que un hoyo puede ser generado consiste en traslapar un q-blob con un conglomerado de al menos dos q-blobs. Dicho q-blob debe traslapar a ambos q-blobs formando el conglomerado (*ver figura 2.3.2 b*).

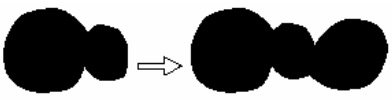


Figura 2.3.2 (a): Lema 1

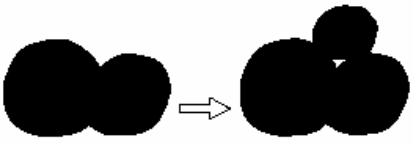


Figura 2.3.2 (b): Lema 2

Teorema 1: sea ΔNB el número de q-blobs y ΔNP el número de puntos de concavidad singulares adicionados a un conglomerado. El número de hoyos adicionales ΔH vale siempre:

$$\Delta H = \frac{\Delta NP}{2} - \Delta NB \tag{2.3.2}$$

Teorema 2: El adicionar dos puntos de concavidad singulares a un conglomerado no altera el número de hoyos.

Teorema 3(a). El número de hoyos para un conglomerado viene siempre dado por:

$$NH = \frac{NP}{2} - NB + 1 \tag{2.3.3}$$

Teorema 3(b). El número de hoyos para NC conglomerados viene siempre dado como:

$$NH = \frac{NP}{2} - NB + NC \tag{2.3.4}$$

Corolario 1. Si NP es el número de puntos de concavidad singulares, NC el número de conglomerados, y NH el número de hoyos en una imagen, entonces el número de q-blobs, NB en una imagen viene dado como:

$$NB = \frac{NP}{2} - NC + NH \tag{2.3.5}$$

La Técnica

La técnica consta de cinco etapas. Cada una de estas se menciona a continuación:

Preprocesamiento de la imagen

En esta etapa una imagen monocromática proveniente de un sensor CCD es primeramente binarizada manualmente.

Determinación del número de conglomerados.

Se obtiene a través del etiquetado de todas y cada una de las componentes conectadas en la imagen binarizada. La imagen es barrida de izquierda a derecha y de arriba a abajo hasta localizar el primer píxel negro, asignándole la primera etiqueta. Una vez encontrado este píxel, sus ocho vecinos son analizados, si cualquiera de éstos es negro es etiquetada con la misma etiqueta del píxel central.

Determinación del número de hoyos en una imagen.

Consiste en invertir la imagen original, contar el número de regiones usando el procedimiento para contar conglomerados, para obtener el valor deseado.

Determinación del número de puntos de concavidad singulares.

La imagen original es primeramente dilatada usando un elemento estructural circular de diámetro siete. La imagen resultante es enseguida erosionada con el mismo elemento estructural. Esto da como resultado una imagen cerrada, morfológicamente hablando. La imagen original es restada de esta imagen cerrada para obtener la imagen final, la cual contendrá tantas regiones conexas como puntos de concavidad singulares haya en la imagen original

Determinación del número de q-blobs en la imagen

El cálculo del número de q-blobs se reduce a aplicar el corolario número 1.

Casos patológicos.

Caso 1. Este caso se presenta cuando al traslaparse dos o más q-blobs, algunos de los puntos de concavidad singulares, no puede ser detectados, debido a las curvaturas de dos q-blobs coinciden (*ver figura 2.3.3.a*).

Caso 2. Este caso se presenta cuando tres o más q-blobs al traslaparse no permiten que el hoyo u hoyos correspondientes se formen (*ver figura 2.3.3.b*).

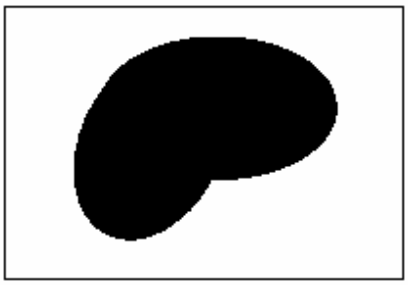


Figura 2.3.3 (a): Caso 1

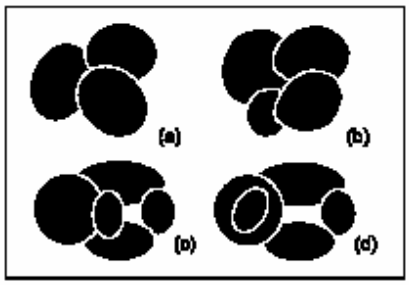


Figura 2.3.3 (b): Caso 2

Ejemplo:

En este caso, la imagen de prueba de la figura 2.3.4.(a) contiene 11, conglomerados, 3 hoyos y 64 puntos de concavidad singulares. Al ser procesada a través del sistema, nos indica que en la imagen hay 40 q-blobs, resultado acertado.

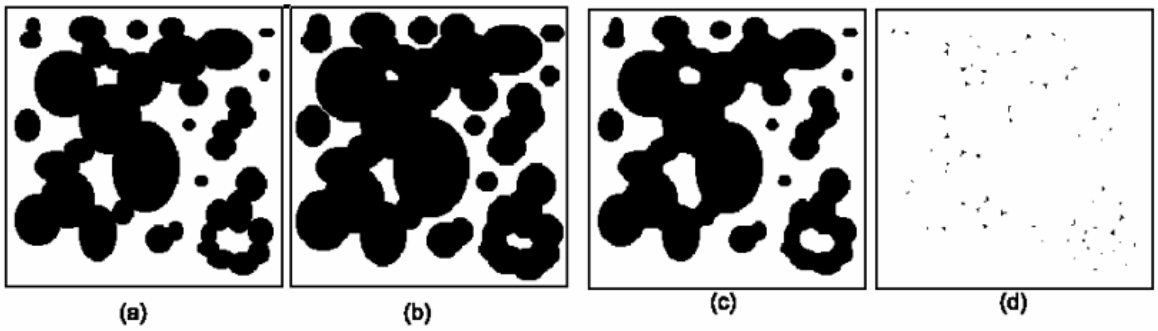


Figura 2.3.4: (a) imagen binaria de prueba, (b) imagen dilatada, (c) imagen erosionada, (d) imagen de concavidades.

Comentarios

Los resultados obtenidos son muy buenos, solamente fallan cuando ocurren casos patológicos o no se determina bien el elemento estructural utilizado.

Capítulo 3

ADQUISICIÓN DE IMÁGENES

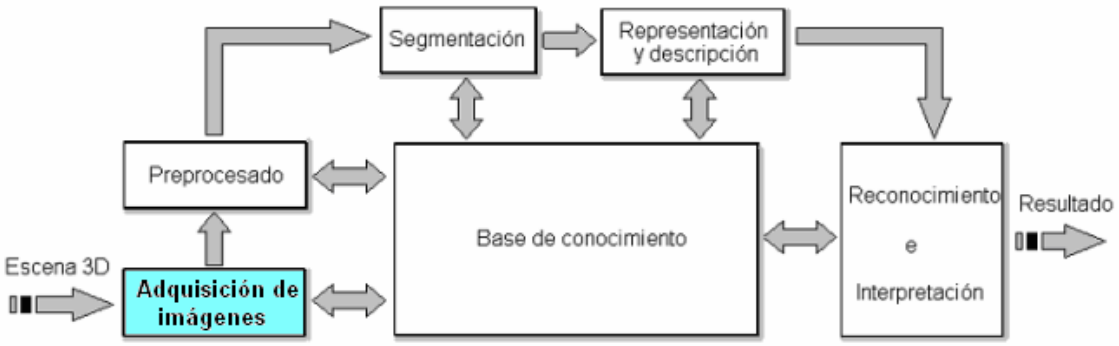


Figura 3.1.0: Etapas fundamentales en el reconocimiento automático de objetivos

En la Figura 3.1.0 se resalta el proceso de Adquisición de imágenes [7]. Este proceso es de gran importancia y determinante para el éxito o fracaso de nuestro sistema de reconocimiento. Se dice que la captación de la imagen es el 50% del sistema y sería un error muy serio pensar que se puede compensar una imagen inadecuada con algún algoritmo; es mejor tener una buena calidad de imagen y utilizar menos algoritmos para su mejoramiento.

El ojo humano tiene casi el mismo funcionamiento que una cámara de vídeo pero no se compara en cuanto a su definición de imágenes y al tiempo de procesamiento, ya que el ojo humano es uno de los órganos más especializados en el reconocimiento. Por todo esto, en este capítulo se menciona la anatomía y el funcionamiento del ojo humano, así como el funcionamiento de la cámara de vídeo, aspectos de iluminación y algunos otros conceptos importantes.

3.1. Elementos de percepción visual

Estructura del ojo humano

El ojo humano tiene la capacidad para captar las imágenes con gran nitidez, el movimiento, la profundidad, el color, el enfoque de cualquier objeto en diferentes distancias, entre otros. Es casi esférico con un diámetro de 20 mm. Es capaz de adaptarse a distintos niveles de iluminación gracias a que el diafragma formado por el iris que puede cambiar de diámetro, proporcionando un agujero central (*la pupila*) que varía entre 2 mm para iluminación intensa y 8 mm para situaciones de poca iluminación.

La luz entra en el ojo a través de la córnea (tejido resistente y transparente que cubre la superficie anterior del ojo) y el iris, atravesando la lente del cristalino hasta llegar a la parte trasera de la esfera ocular llamada retina que está recubierta por una capa de células fotosensibles la cual recibe una pequeña imagen invertida de ese mundo exterior. La lente del cristalino altera su forma para enfocar la imagen [32]. En la figura 3.1.1 se muestran los principales elementos que conforman el ojo humano.

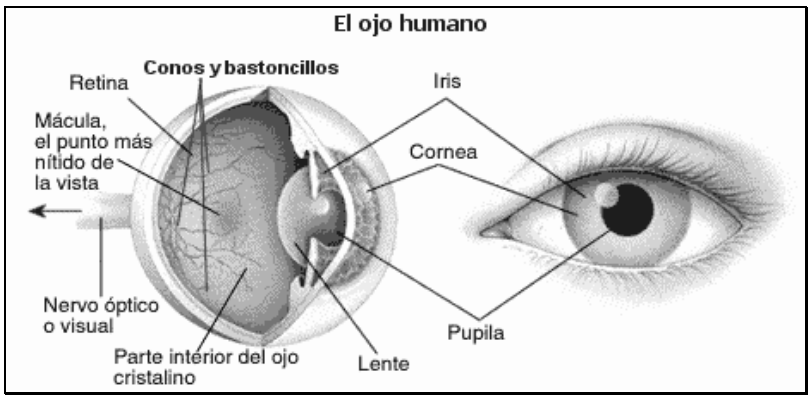


Figura 3.1.1: Anatomía del ojo humano

Para enfocar objetos distantes, los músculos de control hacen que el cristalino se aplane relativamente. De forma similar, estos músculos hacen el cristalino más grueso cuando se enfocan objetos cercanos al ojo.

La distancia entre el punto focal del cristalino y la retina varía entre 17mm y 14mm cuando el poder refractivo del cristalino aumenta desde su mínimo hasta su máximo. Cuando el ojo enfoca a un objeto distante a más de 3 m, el cristalino se coloca en su posición de menor poder refractivo, y cuando el ojo enfoca a un objeto más cercano el cristalino es más fuertemente refractivo [12]. En la figura 3.1.2, el observador esta mirando una palmera de 15m. de altura que se encuentra a 100m. de distancia, si "x" es el tamaño en milímetros de la imagen en la retina, para determinar el valor de "x". Por semejanza de triángulos tenemos:

15/100 = x/17, por lo tanto x=2.25mm.

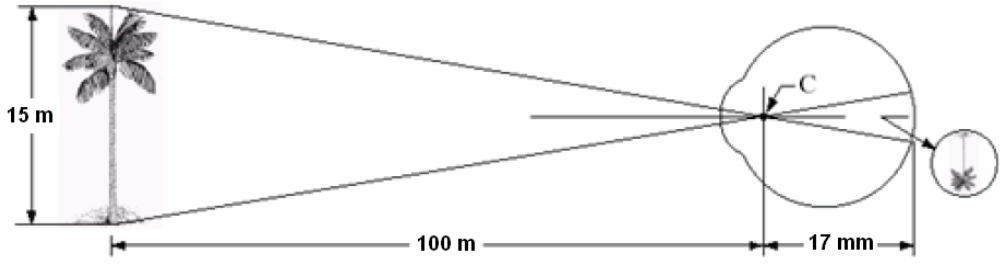


Figura 3.1.2: Formación de imágenes en el ojo humano

En la retina existen dos tipos de receptores: los conos y los bastones (ver figura 3.1.3). En cada ojo existen entre 6 y 7 millones de conos localizados principalmente en la región central de la retina, denominada fóvea, y que son muy sensibles al color. La visión mediante los conos se denomina fotópica o visión de luz brillante. El número de bastones es de 75 a 150 millones que se encuentran distribuidos sobre la superficie de la retina. Los bastones no están implicados en la visión en color y son sensibles a niveles de iluminación bajos. Por ejemplo, los objetos que aparecen con colores brillantes bajo la luz del día, cuando se ven en la luz de la luna aparecen como formas sin color puesto que sólo se estimulan los bastones. Este fenómeno se conoce como visión tenue o visión escotópica.

La figura 3.1.3. nos muestra la densidad de bastones y conos para una sección transversal del ojo derecho cruzando la región por donde emerge el nervio óptico. La ausencia de receptores en esta zona produce lo que se denomina punto ciego. Normalmente no percibimos el punto ciego ya que al ver un objeto con ambos ojos la parte del mismo que incide sobre el punto ciego de uno de ellos, incide sobre una zona sensible del otro. Si cerramos un ojo tampoco seremos consientes de la existencia del punto ciego debido a que el cerebro normalmente nos engaña y completa la parte que falta de la imagen [12].

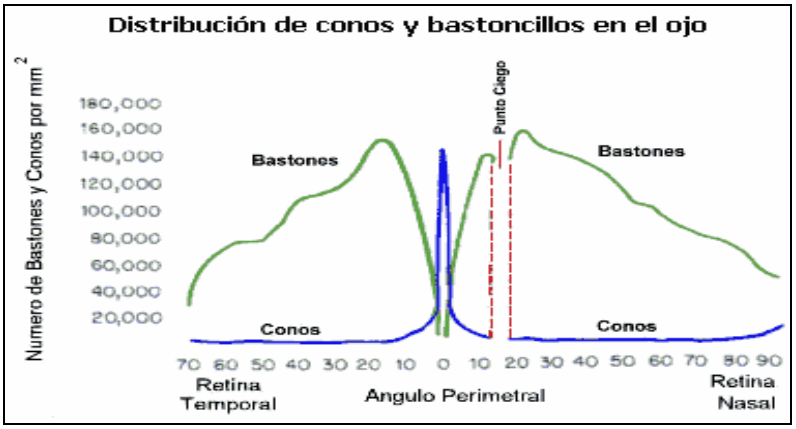


Figura 3.1.3: Distribución de conos y bastones en la retina

Percepción de color

Como se ha mencionado, el ser humano percibe el color mediante los sensores (*conos*) que traducen la energía lumínica en señales nerviosas. Estos se pueden dividir en tres clases, dependiendo de la banda de longitudes de onda a la cual son más sensibles: Los sensores tipo alfa(α) tienen una mayor sensibilidad a 480 nanómetros (*azul*), los tipo beta(β) a 540 nm. (*verde*) y los tipo gama(γ) a 570 nm. (rojo) [33], como se aprecia en la figura 3.1.4.

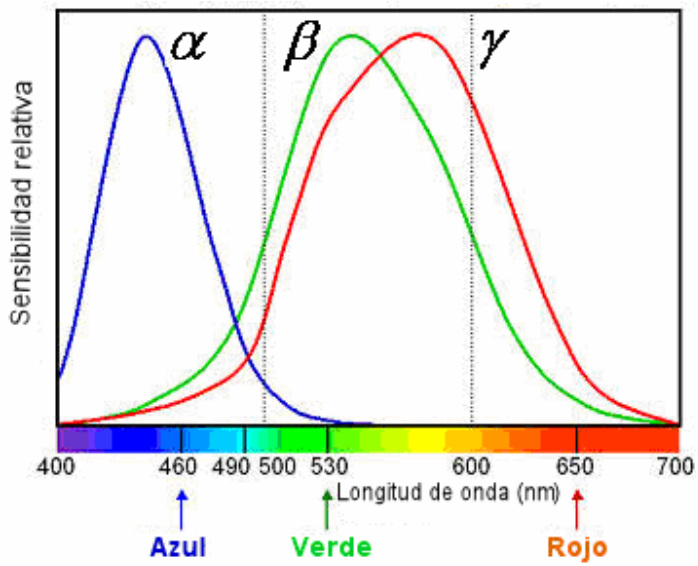


Figura 3.1.4: Sensibilidad espectral del ojo humano.

La identificación del color en la imagen se hace mediante la combinación de estas tres señales, de donde se perciben la gran variedad de colores que podemos distinguir. A estos se les denomina colores primarios (*rojo, verde y azul*). De la combinación aditiva en pares iguales de éstos, obtenemos los colores secundarios (*amarillo, magenta y cyan*); y de la combinación de los tres obtenemos el blanco. Al combinar los secundarios substractivamente obtenemos los colores primarios y el negro [13] (*figuras 3.1.5 y 3.1.6*).

Figura 3.1.5: Mezclas de luz (adición de primarios)

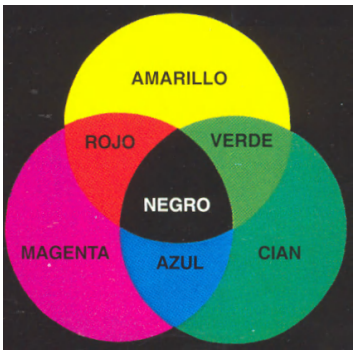


Figura 3.1.6: Mezclas de pigmentos (substracción de secundarios).

3.2 Captación de la imagen por la cámara

La captación de la imagen se realiza por medio de sensores que se encuentran dentro de la cámara. Estos sensores son componentes sensibles a la luz que modifican su señal eléctrica en función de la intensidad luminosa que perciben.

La tecnología más habitual en este tipo de sensores es el CCD (dispositivos de acoplamiento de carga) donde se integra en un mismo chip los elementos fotosensibles y el conjunto de puertas lógicas y circuitería de control asociada (*ver figura 3.2.1*). En éstos dispositivos, la señal eléctrica que transmiten los fotodiodos es función de la intensidad luminosa que reciben, su espectro, y el tiempo de integración.

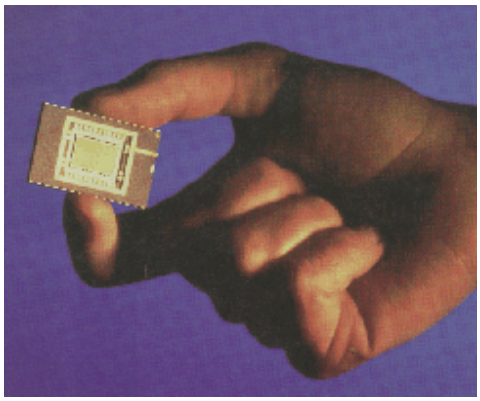


Figura 3.2.1: Dispositivo de acoplamiento de carga

Otra tecnología que está empezando a extenderse son los sensores CMOS (*complementary metal oxide semiconductor*) dada las ventajas de éstos sobre los CCD, y la reducción de precios de estos dispositivos. En cuanto al rango dinámico se pasa de los 70 decibeles de los sensores CCD a los 120dB de los sensores CMOS, valor más cercano a los 200dB del ojo humano, lo que facilita la autoadaptación en el propio chip al brillo existente en el entorno.

Existen diferentes arquitecturas de sensores empleados en los dispositivos de captura. Básicamente, hay dos modelos: lineales o de barrido y sensores de área.

Los sensores lineales son una línea de fotodiodos que se van desplazando para leer la imagen. Este tipo de tecnología se emplea únicamente en escáneres y en algunos respaldos digitales. Esta arquitectura permite la utilización de sensores de 1x1024, 1x2048, 1x4096, e incluso 1x6000 píxeles, lo que la hace muy adecuada para trabajar con altas resoluciones [34], (ver figura 3.2.2).

En segundo lugar están los sensores de área. Estos alcanzan resoluciones habituales de 1024x1024. Son este tipo de sensores los que emplean todas las cámaras digitales, tanto las réflex como las compactas. Entre los más comunes encontramos los siguientes:

Sensores CCD/CMOS: Delante del sensor digital se coloca un filtro RGB, el llamado mosaico de Bayer. Así, cada fotodiodo sólo recibe información de uno de los tres colores y los otros dos canales de cada píxel interpolan. Evidentemente, se trata de una operación muy sofisticada, donde cada píxel es capaz de procesar la información de las células vecinas mediante algoritmos y "adivinar" así cuáles serían los dos valores que le faltan.

Sensores Foveon x3: Son los únicos sensores del mercado que capturan el color real de las imágenes sin necesidad de inventar ninguno de los tres canales (ver figura 3.2.3). De esta manera, se evita problemas tan clásicos como el moiré que son una serie de puntos superpuestos que crearán una ilusión de más colores. Al escanear una foto, usualmente suele aparecer un patrón Moire. Esto suele ocurrir por incorrectos ángulos de pantalla de medios tonos sobreimpresionados.

Figura 3.2.2: Sensores lineales

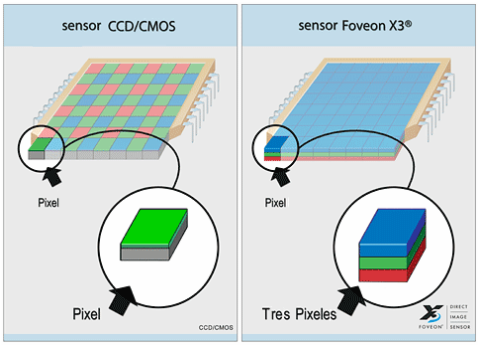


Figura 3.2.3: Sensores de área

3.3. Iluminación

La iluminación es el aspecto más decisivo de cualquier aplicación de visión artificial, si el objeto tiene una buena iluminación se pueden lograr muy buenos resultados en nuestro sistema de reconocimiento de objetos. Muchas aplicaciones buenas han fallado por la falta de una iluminación apropiada. En un sistema de visión artificial, la mejor imagen es aquella que tiene mayor contraste, donde las áreas de interés se destacan del fondo (*background*) que no tiene mayor importancia.

La luz es reflejada de dos maneras llamadas reflexión especular y reflexión difusa. En la reflexión especular, cada rayo incidente se refleja en una única dirección como en el caso del algunos metales o el espejo, que pueden saturar el sensor de la cámara, en este caso lo mejor es utilizar alguna técnica de iluminación especial. Las reflexiones difusas son tenues pero estables, la intensidad de reflexión puede ser de 10 a 1000 veces menor que la fuente de luz [14], *ver figuras 3.3.1 y 3.3.2*.

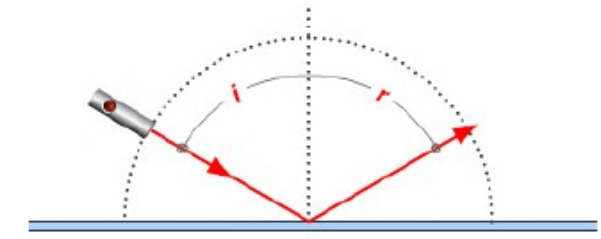


Figura 3.3.1: Reflexión Especular

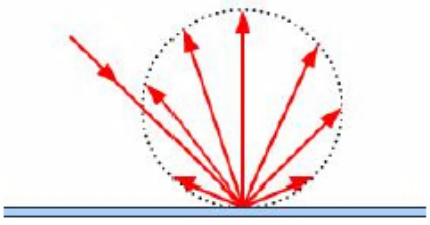


Figura 3.3.2: Reflexión difusa

3.4. Resolución de la imagen

Para las imágenes digitales almacenadas como mapa de bits, la resolución de la imagen describe cuán nítida es una imagen de fotografía por medio de dos números enteros, donde el primero es la cantidad de columnas de píxeles (*cuántos píxeles tiene la imagen a lo ancho*) y el segundo es la cantidad de filas de píxeles (*cuántos píxeles tiene la imagen a lo alto*). En la figura 3.4.1 se presenta una ilustración de una imagen con diferente resolución.

Figura 3.4.1: Diferentes tipos de resolución .

La imagen capturada a 160x200 píxeles pierde detalles del objeto a reconocer (figura 3.4.1); por el contrario, la imagen capturada a 640x480 píxeles es muy nítida, pero con el inconveniente de que contiene mucho más información y su procesamiento es más lento.

3.5. Formatos de imágenes

Para almacenar imágenes en forma digital las compañías de diseño lo hacen de diferente manera (*formato*) de acuerdo a sus necesidades: calidad de la imagen, compresión, número de imágenes en un solo archivo, entre otras; es por esto que al momento de intentar abrir un archivo de imagen, nos damos cuenta que no se puede leer o que la imagen es de mala resolución. Para almacenar fotografías con una buena calidad se utilizan los formatos *.bmp* y *.tiff*, pero tienen la desventaja de que ocupan 10 veces o más que un *jpg (jpeg)*. Ver figura 3.5.1

Figura 3.5.1: Tamaño de una misma imagen con diferentes tipos de formatos.

A continuación se muestran algunos de los formatos más comunes y las posibilidades que brinda cada uno:

BMP.- El formato BMP (*Bit Map*) es el formato de las imágenes en bitmap de Windows. Aunque muy extendido, tiene la dificultad de la escasa compresión que realiza en los archivos por lo que ocupan mucho espacio. Pero el formato de Mapa de Bits tiene una importante característica a su favor, es que casi todos los usuarios tienen una *PC* con ambiente gráfico Windows y pueden leerlo desde un accesorio llamado "*Paint"*.

TIFF.- El formato *TIFF*, que corresponde a las siglas *Tagged-Image File Format*, se utiliza cuando se van a realizar impresiones en papel de la imagen. Es un formato que admite una compresión muy baja, por lo que la pérdida en la calidad de imagen es prácticamente nula. Se conoce como formato de compresión sin pérdida. Una desventaja es que los archivos .tiff son de gran tamaño.

GIF.- Estas tres letras, representan las siglas de *"Graphics Interchange Format"*. La tecnología del GIF fue desarrollada por la empresa CompuServe, destacando en sus características la posibilidad de trabajar, con un máximo de 256 colores y con mas de una imágen, el GIF es un formato ideal para utilizar en la web en imágenes pequeñas o de pocos colores y no es recomendable para utilizar en impresión, ya que la calidad es limitada.

JPEG o JPG.- El formato JPEG(*Joint Photographic Experts Group*) está diseñado para realizar compresión de imágenes, permitiendo reducir la cantidad de información de las mismas, con una consecuente reducción de peso del archivo final. Es por ello que dicho formato también es de uso común en la web. La resolución de este formato es baja y no es recomendable para la impresión.

PCX.- Este es el formato desarrollado por *Zsoft* para su programa Paintbrush para PC. En un principio sólo guardaba 16 colores, pero las últimas actualizaciones acomodan el color de 8 y de 16 bits.

Capítulo 4

PREPROCESAMIENTO DE LA IMAGEN

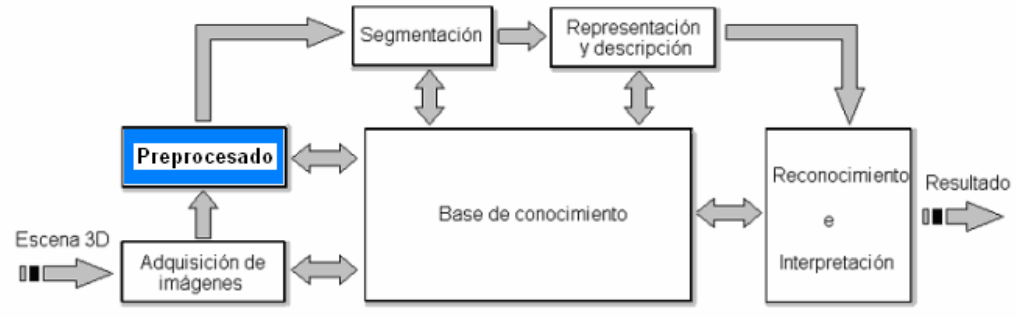


Figura 4.1.0: Etapas fundamentales en el reconocimiento automático de objetivos

En la Figura 4.1.0 se resalta el segundo paso para el reconocimiento automático de objetivos: el *"Preprocesado"* o procesamiento de la imagen.

Cuando se adquiere una imagen mediante cualquier sistema de captura, por lo general ésta no se utiliza directamente por el sistema de visión. La aparición de variaciones en intensidad debidas al ruido, por deficiencias en la iluminación o la obtención de imágenes de bajo contraste hace necesario un preprocesamiento de la imagen con el objetivo fundamental de corregir estos problemas, además de aplicar aquellas transformaciones a la imagen que acentúen las características que se deseen extraer de las mismas, de manera que se facilite las operaciones de las etapas posteriores.

En el presente capítulo se mencionan las diferentes técnicas empleadas para el mejoramiento de la imagen y las transformaciones necesarias para la aplicación de estas técnicas.

4.1. Conversión de una imagen en color RGB a niveles de gris

En el proceso de captación de imagen, obtuvimos una imagen digital en el espacio de color RGB con sus tres matrices de intensidades para los colores rojo, verde y azul respectivamente (*ver capítulo 6*). La figura 4.1.1 nos muestra las tres matrices correspondientes únicamente al recuadro de la imagen con intensidades para cada color que van del 0 al 255.

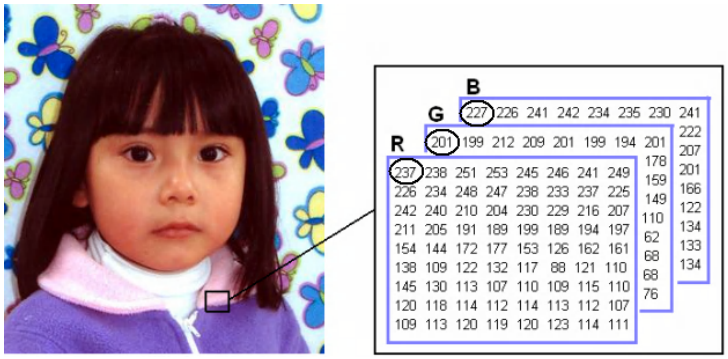


Figura 4.1.1: Representación de una imagen en color con sus tres matrices de intensidades.

El proceso de conversión a niveles de gris consiste en calcular el promedio de cada intensidad para las matrices de los colores rojo, verde y azul . Cada valor de la conversión se redondea para formar una matriz de intensidades donde el valor correspondiente a cada punto indica el valor de intensidad que puede ir de [0 a 255]; el cero representa el negro absoluto y el 255 el blanco absoluto. La figura (4.1.2.) se obtuvo como resultado de aplicar a la figura (4.1.1.) la siguiente fórmula:

$$I = Round\left\{\frac{1}{3}(R+G+B)\right\} \qquad (4.1.1)$$

Por ejemplo, para el cálculo de los dos primeros elementos de la matriz en niveles de gris tenemos:

$$I_{11} = Round\left\{\frac{1}{3}(237+201+227)\right\} = Round\{221.\bar{6}\} = 222$$

$$I_{12} = Round\left\{\frac{1}{3}(238+199+226)\right\} = Round\{221.0\} = 221$$

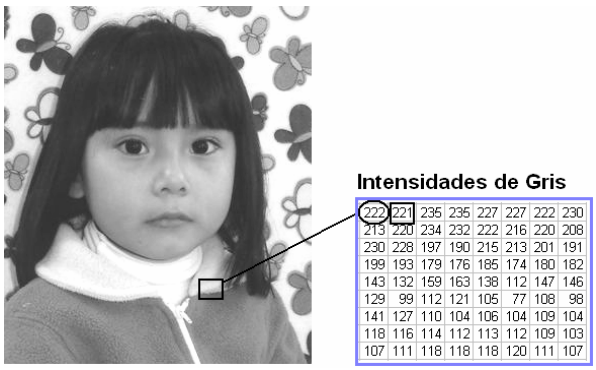


Figura 4.1.2: Imagen es escala de gris con su matriz de intensidades.

4.2. Conversión de una imagen en niveles de gris a blanco y negro (Binarización)

Con esta técnica, los valores de píxel en la imagen de entrada menores a un cierto umbral (*entre 0 y 255*) son convertidos a negro mientras que los píxeles con valores mayores o iguales al umbral son convertidos a blanco.

En la figura 4.2.1 se muestra la transformación de la imagen en escala de gris a blanco y negro. El valor de la intensidad mayor o igual a un umbral (127) se les asigna el máximo valor de gris (*255=blanco*). Por su parte, los píxeles comprendidos por debajo de este valor toman el valor mínimo (*0=negro*).

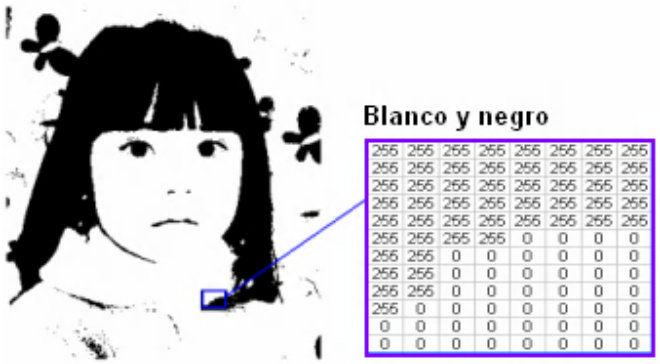


Figura 4.2.1: imagen binarizada y su matriz de intensidades.

4.3. Mejoramiento de histograma

El histograma de una imagen es una función que representa el número de píxeles existentes para cada valor de nivel de gris. Para una imagen con 256 niveles de gris posibles, la abscisa del histograma varía entre 0 y 255 (*figura 4.3.1*). La probabilidad "P_i" de ocurrencia de un determinado nivel "i" se define como:

$$P_i = \frac{f_i}{N * M} \qquad (4.3.1)$$

Donde "f_i" es el número de píxeles en el nivel de intensidad "*i*" y "*N*M*" es la cuenta total de píxeles correspondientes a las filas y columnas de la imagen. Todos los valores de "P_i" son menores o iguales que "1" y la suma de todos los valores de "P_i" es 1.

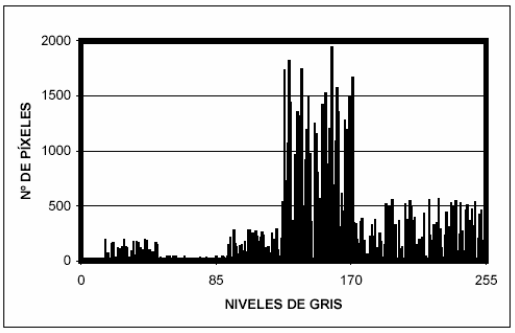


Figura 4.3.1: Histograma de una imagen en escala de gris

Un histograma con la gráfica hacia la izquierda resulta muy obscura, si está hacia la derecha contiene mucho brillo y si se encuentra al centro tiene poco contraste. La representación de un histograma ideal seria la de una recta horizontal, ya que eso nos indicaría que todos los posibles valores de grises están distribuidos de manera uniforme en nuestra imagen (*figura 4.3.2.*).

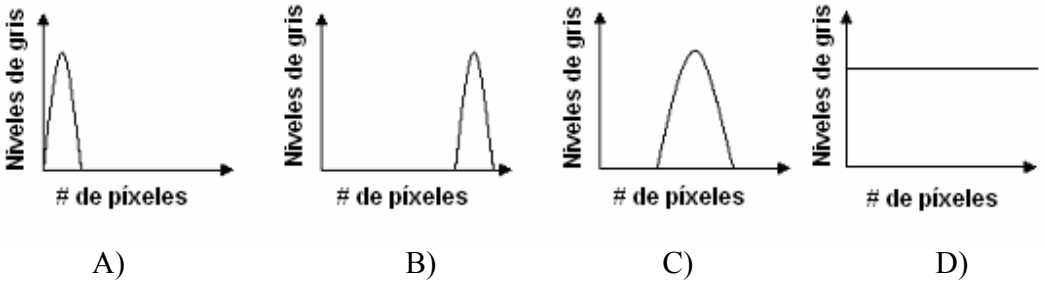


Figura 4.3.2: Diferentes tipos de histogramas: A) imagen muy obscura, B) imagen muy clara, C) imagen con poco contraste, D) histograma ideal.

Desplazamiento de histograma

El desplazamiento de histograma se usa para aclarar u oscurecer una imagen pero manteniendo la relación entre los valores de los niveles de gris. Esta operación puede llevarse a cabo por simple adición o sustracción de un número fijo a todos los niveles de gris: $g(x,y) = f(x,y) \pm desplazamiento$

Las figuras 4.3.5 y 4.3.6 muestran la imagen resultante y su respectivo histograma tras aplicar un desplazamiento del histograma 100 unidades a la derecha para aclarar la imagen de la figura 4.3.3.

Figura 4.3.3: Imagen obscura

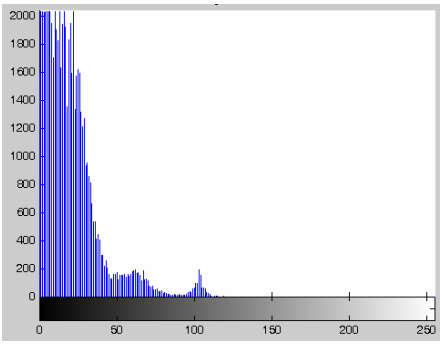


Figura 4.3.4: Histograma de la imagen obscura.

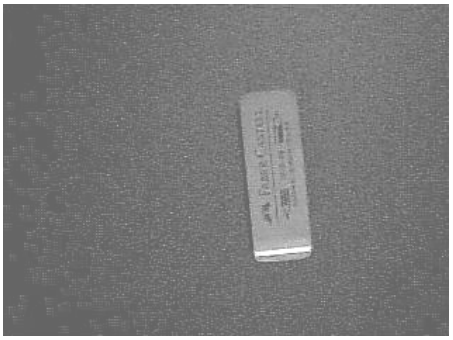


Figura 4.3.5: Imagen con desplazamiento

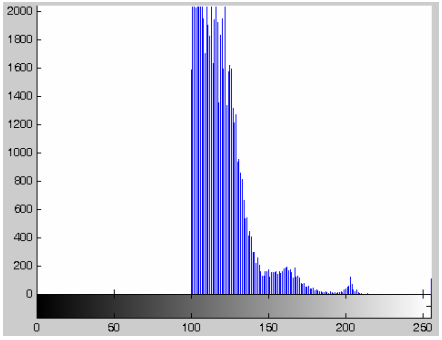


Figura 4.3.6: Histograma con desplazamiento.

Ampliación de histograma

Es una de las técnicas más utilizadas para la mejora del contraste de la imagen mediante una determinada transformación o modificación del histograma. Se trata de encontrar una función *F(g)* que realce el contraste de la imagen original expandiendo la distribución de los niveles de gris. Dicha expansión debe ser lo más suave posible en el sentido de que idealmente debería haber el mismo número de píxeles por niveles de gris.

Como ejemplo utilizaremos la siguiente función analítica,

$$F(g) = \max\left\{0, round\left(\frac{N_g \, x F_a}{N \, x M}\right) - 1\right\} \qquad (4.3.2)$$

donde "*Fa"* es el número total de píxeles en el nivel g más los inferiores, es decir es la frecuencia acumulativa para los niveles de g e inferiores. Si consideramos una imagen de 70 píxeles (*NxM* = 70) y 16 niveles de intensidad (*Ng*=16) con valores en el rango [0...15], "*g"* son los niveles de gris, *"f"* es la frecuencia de aparición de dichos niveles de gris, Fa es la frecuencia acumulativa de los niveles de gris y *F(g)* es el resultado de la transformación final. En las figuras (4.3.7.) y (4.3.8.) se muestra el histograma correspondiente a la tabla (4.3.1.) antes y después de la transformación.

g	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
F	3	5	10	12	12	9	7	2	6	4	0	0	0	0	0	0
F_a	3	8	18	30	42	51	58	60	66	70	70	70	70	70	70	70
F(g)	0	1	3	6	9	11	12	13	14	15	15	15	15	15	15	15

Tabla 4.3.1: Distribución de los niveles de gris antes y después de la igualación

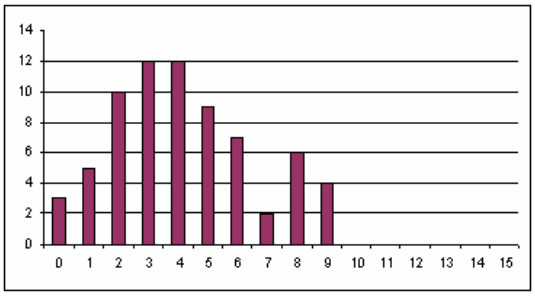


Figura 4.3.7: Histograma correspondiente a la tabla 4.3.1 antes de la ampliación.

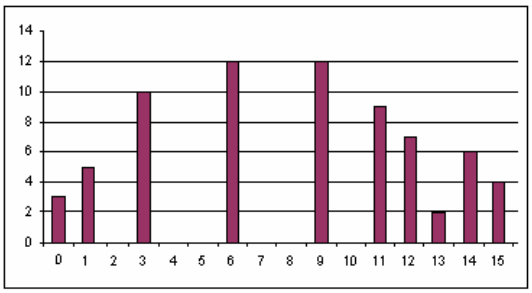


Figura 4.3.8: Histograma obtenido después de la ampliación

En las figuras (4.3.9.) y (4.3.10.) se implementó este método de ampliación de histograma para mejorar el contraste de la imagen de la figura (4.3.3.). Como nos damos cuenta, se resalta el objeto a reconocer, pero también resalta el ruido producido por reflejos de luz y por el sistema de captación de la imagen. Es muy importante que posteriormente realizar un buen filtrado y una adecuada segmentación para que los resultados en el sistema de reconocimiento sean los adecuados.

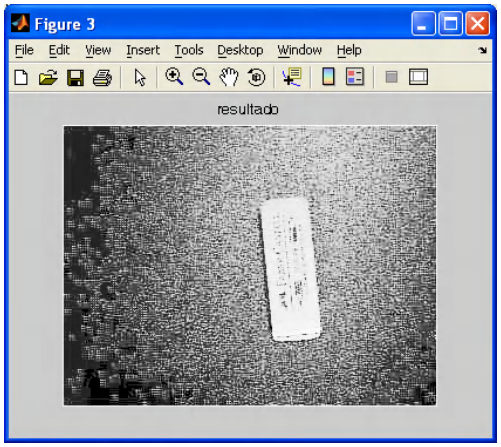


Figura 4.3.9: Imagen obtenida por el método de ampliación de histograma.

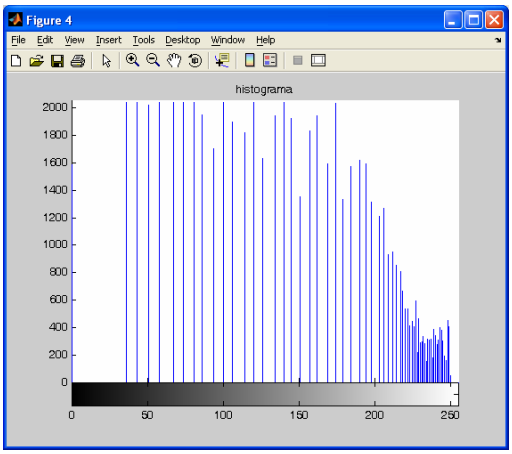


Figura 4.3.10: Histograma correspondiente a la ampliación de histograma

4.4. Uso de filtros para el mejoramiento de la imagen

En este proceso se trata de reducir al máximo el ruido contenido en la imagen producido por el tipo de iluminación, las sombras entre los objetos, pequeños puntos no deseados dentro de la imagen y otros efectos que pueden estar presentes en una imagen digital como resultado del muestreo, cuantización, transmisión o por perturbaciones en el sistema como pueden ser las partículas de polvo en el sistema óptico. Para la eliminación de todo este tipo de ruido se emplean diferentes tipos de filtros.

Filtrar una imagen (*f*) consiste en aplicar una transformación (*T*) para obtener una nueva imagen (*g*) de forma que ciertas características son acentuadas o disminuidas:

$$g(x,y) = T\left[f\left(x,y\right)\right] \qquad (4.4.1)$$

Se puede considerar que la señal (*imagen*) pasa a través de una caja (*filtro*) cuya salida es la imagen filtrada (*figura 4.4.1*).

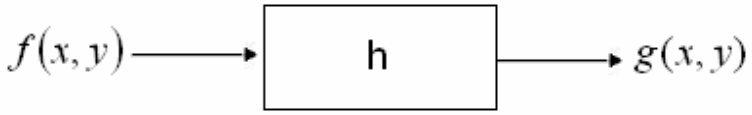


Figura 4.4.1: Proceso de filtrado

De acuerdo a la teoría de sistemas, al pasar una señal por un sistema lineal, la salida es la convolución de la transformación del sistema (*función de transferencia*) con la señal de entrada:

$$g(x,y) = h(x,y) * f(x,y) \qquad (4.4.2)$$

Filtrado en el dominio espacial

Las técnicas de filtrado en el dominio espacial operan directamente sobre los píxeles de la imagen mediante una mascara cuadrada o rectangular. Una máscara es una pequeña imagen que consiste en una serie de valores predeterminados para cada posición (*ver figura 4.4.2*). La máscara se centra sobre el píxel de interés, de tal forma que el nuevo valor del píxel depende de los píxeles que cubren la máscara.

$w_{1,1}$	$w_{2,1}$	$w_{3,1}$
$w_{2,1}$	$w_{2,2}$	$w_{3,2}$
$w_{3,1}$	$w_{2,3}$	$w_{3,3}$

Figura 4.4.2: Ejemplo de máscara de 3x3

A cada celda de la mascara le corresponde un peso o coeficiente (w), donde el nuevo valor del píxel es la sumatoria del producto de los píxeles vecinos con el peso correspondiente:

$$g(x,y) = \sum_x \sum_y f(x,y)w(x,y) \qquad (4.4.3)$$

Filtro promedio aritmético

Sea w_{xy} un conjunto de coordenadas en una subimagen cuadrada(*ventana*) de tamaño n^2 centrada en el punto $p(x,y)$, como se muestra en la figura 4.4.3 . El filtro promedio aritmético calcula el valor promedio de la imagen con ruido en el área w_{xy}. El valor de la imagen restaurada $\hat{g}$ en el punto $p(x,y)$ es el promedio aritmético calculado en esa vecindad.

$$\hat{g}(x,y) = \frac{1}{n^2} \sum_{(x,y)\in w_{xy}} f(x,y) \qquad (4.4.4)$$

x-1,y-1	x,y-1	x+1,y-1
x-1,y	x,y	x+1,y
x-1,y+1	x,y+1	x+1,y+1

Figura 4.4.3: conjunto de coordenadas de una mascara de 3x3 (w_{xy}) para el punto $p(x,y)$

El valor de la intensidad en el punto(x,y) será el resultado de la ecuación anterior haciendo un barrido sobre toda la imagen original iniciando en el segundo renglón y segunda columna y terminando en el renglón n-1 y columna n-1 aplicando la siguiente mascara:

$$p = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad (4.4.5)$$

Por ejemplo, si tenemos la siguiente mascara:

$$p = \frac{1}{9}\begin{bmatrix} 39 & 46 & 41 \\ 10 & 33 & 49 \\ 72 & 77 & 36 \end{bmatrix}$$

el valor del punto(x,y) de la imagen restaurada ($\hat{g}$) es igual a la suma de los 9 valores de la matriz divididos por 9, el resultado correspondiente haciendo el redondeo al entero más próximo para el punto “p” será de 45. Como se puede observar en las figuras 4.4.4 y 4.4.5, el filtro promedio tiene el inconveniente de desdibujar bordes y formas; pero los puntos oscuros o con mucho brillo se atenúan.

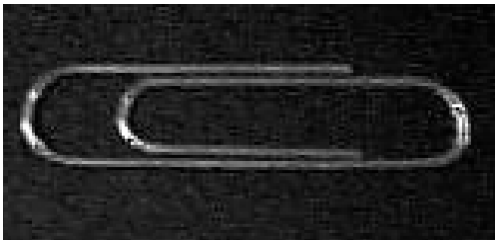


Figura 4.4.4: imagen en niveles de gris.

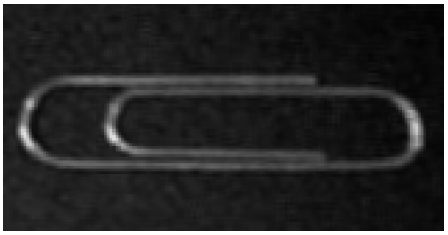


Figura 4.4.5: imagen resultado al aplicar un filtro promedio

Filtro máximo, mínimo y mediana

El filtro Máximo consiste en asignar al píxel de referencia el valor más alto de los encontrados en la máscara, por lo que su efecto es ensanchar las zonas claras y adelgazar las oscuras. El filtro Mínimo hace justo lo contrario, seleccionando el valor más bajo. En el Filtro Mediana se escoge el píxel de en medio del vector (ordenado previamente de menor a mayor). En las figuras 4.4.6, 4.4.7. y 4.4.8 se muestran los resultados obtenidos al aplicar estos tipos de filtros a la imagen de la figura 4.5.4. Sus ecuaciones son las siguientes:

$$p\_\max = \max\{f1, f2, ..., fn\} \quad (4.4.6)$$

$$p\_\min = \min\{f1, f2, ..., fn\} \quad (4.4.7)$$

$$p\_med = med\{f1 \le f2 \le f3 \le ...fn\} \quad (4.4.8)$$

En la ecuación (4.4.6) y ecuación (4.4.7) se colocan en un vector los nueve valores y el valor del punto(x,y) de la imagen restaurada $\hat{f}$ será el máximo o el mínimo de estos valores, en la ecuación (4.4.8) la mediana se escoge del píxel de en medio. Ejemplos:

$$p\_\max = \max\{39,46,41,10,33,49,72,\underline{77},36\} = \mathbf{77}$$

$$p\_\min = \min\{39,46,41,\underline{10},33,49,72,77,36\} = \mathbf{10}$$

$$p\_med = med\{10,33,36,39,\underline{41},46,49,72,77\} = \mathbf{41}$$

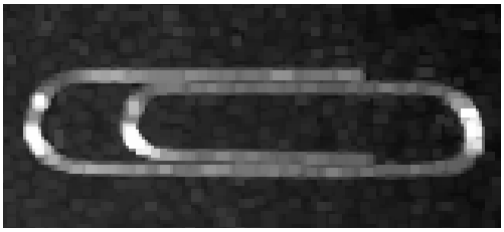


Figura 4.4.6: filtro "punto máximo"

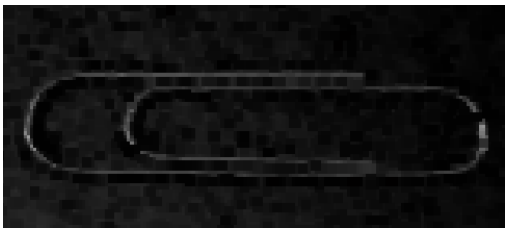


Figura 4.4.7: filtro "punto mínimo"

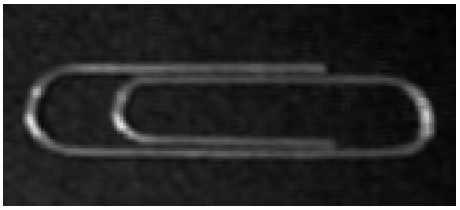


Figura 4.4.8: filtro "mediana"

4.5. Suavizado binario

Al binarizar una imagen se producen pequeños huecos, esquinas perdidas, puntos aislados y contornos irregulares (ruido). Para reducir este tipo de ruido se utilizó una expresión booleana específica sobre un entorno de vecindad centrado sobre un píxel "*p*" y dependiendo de la configuración espacial y los valores binarios de sus vecinos se le asigna al píxel "*p*" el valor de "0" o un "1", en este caso se utiliza la siguiente la mascara de 3x3 de la figura 4.5.1 y para simplificación se utiliza la mascara de la figura 4.5.2, donde se asigna una letra a cada píxel de la mascara.

x-1,y-1	x,y-1	x+1,y-1
x-1,y	x,y	x+1,y
x-1,y+1	x,y+1	x+1,y+1

Figura 4.5.1: Coordenadas del punto p(x,y) y sus vecinos

$$\begin{bmatrix} a & b & c \\ d & p & e \\ f & g & h \end{bmatrix}$$

Figura 4.5.2: Asignación de letras a cada vecino del punto "p"

Para rellenar los pequeños huecos de un píxel en zonas oscuras y eliminar los pequeños cortes y muescas en segmentos de lados rectos se utiliza la siguiente expresión booleana:

$$x1 = p + b * g * (d + e) + d * e * (b + g) \qquad (4.5.1)$$

donde el signo "+" y el "*" representan las funciones lógicas AND y OR respectivamente. Si "*x1"* toma el valor de 1, asignamos un "*1*" a *p* y en caso contrario este píxel tomará el valor de "0". El resultado de la ecuación anterior se almacena en una nueva imagen para que el resultado de las operaciones no afecte a la imagen original (*figura 4.5.3*).

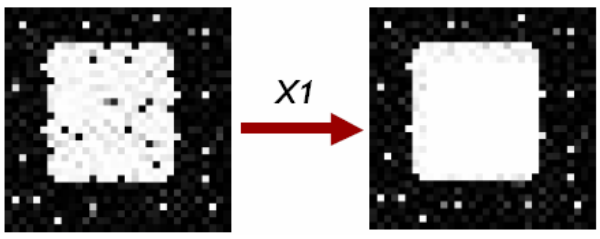


Figura 4.5.3: Eliminación de pequeños huecos.

Si queremos eliminar los unos aislados y las pequeñas protuberancias a lo largo de los segmentos de lados rectos (*ver figura 4.5.4*), se utiliza la siguiente expresión booleana:

$$x2 = p*[(a+b+d)*(e+g+h)+(b+c+e)*(d+f+g)] \qquad (4.5.2)$$

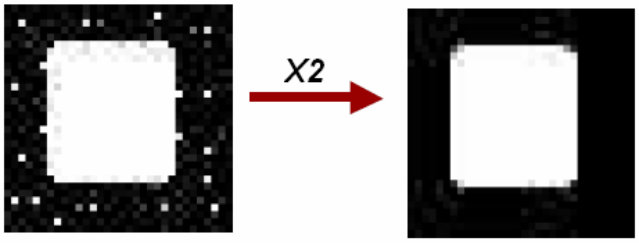


Figura 4.5.4: Eliminación de pequeñas protuberancias.

4.6. Operaciones lógicas sobre imágenes en blanco y negro

Las operaciones lógicas han sido ampliamente usadas en visión por computadora para la detección de rasgos y análisis de imágenes. Las principales operaciones lógicas entre imágenes son *AND, OR* y *NOT*. Las operaciones se realizan entre píxeles de imágenes binarias. De esta forma, si *p1* y *p2* son respectivamente los píxeles de dos imágenes en cuestión, entonces:

P1	**P2**	P1 **AND** P2	P1 **OR** P2
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Tabla 4.6.1: Operaciones binarias AND y OR.

Como se aprecia en la tabla anterior, el resultado de la operación *AND* es "*1*" cuando los dos píxeles de las imágenes en cuestión son "*1*" y en la operación *OR* basta con que alguno de los dos píxeles sea "*1*" para que el resultado sea "*1*". La operación *NOT* se lleva a cabo con los píxeles de una sola imagen y el resultado es el complemento o la negación del píxel en cuestión:

P1	**NOT** P1
0	1
1	0

Tabla 4.6.2: Operación NOT.

Existen mas operaciones que se derivan de estas operaciones básicas, por ejemplo: *NAND*, *NOR* y *XOR*. La figuras 4.6.1 ejemplifica las operaciones básicas aplicadas a imágenes binarias.

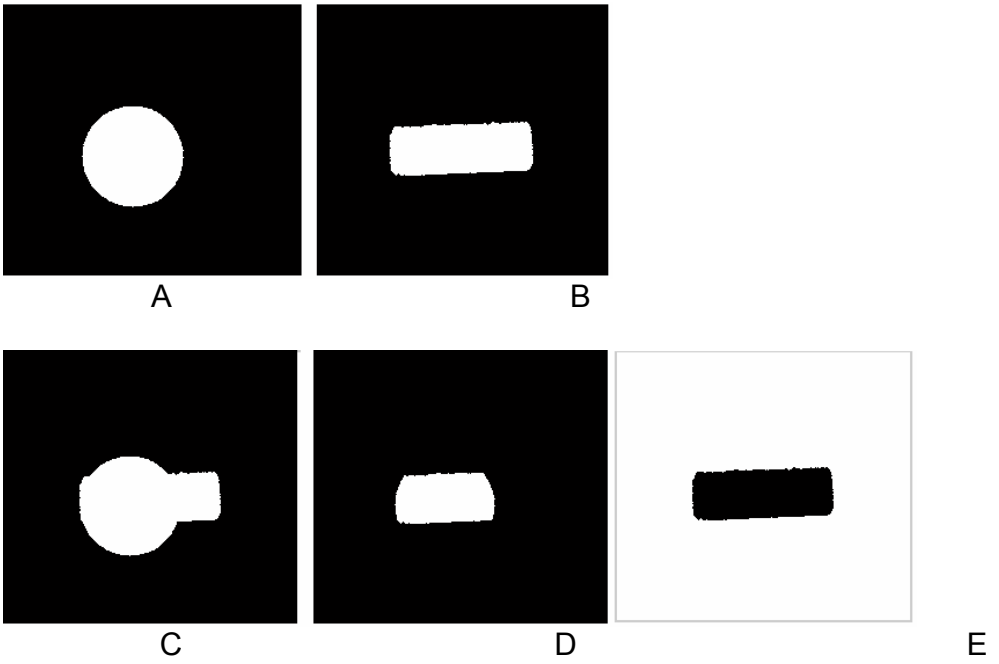


Figura 4.6.1 Aplicación de operaciones binarias básicas: A) imagen "A", B) imagen "B", C) *C = A or B*, D) *D = A and B*, E) *E = not B*.

4.7. Operaciones aritméticas sobre imágenes en niveles de gris

Las operaciones aritméticas han sido ampliamente utilizadas en el procesamiento digital de imágenes. Las más comunes son:

Operación	Fórmula	
Suma	$(p1 + p2)/2$	(4.7.1)
Resta	$abs(p1 - p2)$	(4.7.2)
Multiplicación	*p1*p2*	(4.7.3)
División	*p1/p2*.	(4.7.4)

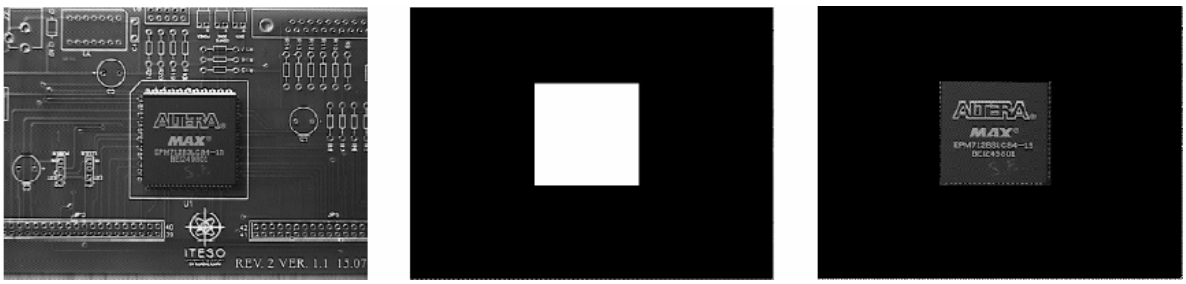
Estas operaciones se realizan entre píxeles de imágenes distintas. De esta manera, *p1* pertenece a una de las dos imágenes y *p2* a la otra.

La operación de suma se utiliza para reducir el ruido en una imagen, también se utiliza para combinar dos o más imágenes. La resta de imágenes ha sido usada extensamente para la detección de movimiento entre imágenes captadas en instantes de tiempo sucesivos, y para remover información estática del fondo. La multiplicación es utilizada para aislar regiones de interés en la imagen, multiplicando una imagen en niveles de gris por una máscara binaria. La división al igual que la multiplicación se aplican principalmente para ajustar el brillo en una imagen o para corregir sombras causadas por iluminación no uniforme durante el proceso de adquisición. En la figuras 4.7.1, 4.7.2 y 4.7.3 se muestran algunos ejemplos de operaciones aritméticas con imágenes [16].

Figura 4.7.1. Efecto de aplicación del operador suma para combinar dos imágenes.

A B C

Figura 4.7.2: Efecto de aplicar la operación de resta para detectar movimiento C = A-B.



A B C

Figura 4.7.3: Efecto de aplicar la multiplicación para aislar regiones de interés C = A*B.

4.8. Operaciones morfológicas básicas

Se utilizan para eliminar pequeños puntos aislados que no pertenecen a nuestro objeto en cuestión mediante ciertas técnicas de morfología matemática como son: *erosión, dilatación, apertura y cerradura.*

Dilatación.- La dilatación $A \oplus B$ es el conjunto de puntos de todas las posibles sumas de pares de elementos uno de cada conjunto A y B:

.

$$A \oplus B = \bigcup_{\beta \in B} (A + \beta) \qquad (4.8.1)$$

Por ejemplo, dados A y B en forma de vectores para los valores correspondientes a los unos cuyas coordenadas se definen con respecto a $[0,0]$.

$$A = \{(0,1),(1,1),(2,1),(2,2),(3,0),(4,0)\}$$
$$B = \{(0,0),(0,1)\}$$
$$A \oplus B = \begin{Bmatrix}(0,1),(1,1),(2,1),(2,2),(3,0),(4,0),\\(0,2),(1,2),(2,2),(2,3),(3,1),(4,1)\end{Bmatrix}$$

En los conjuntos A y B, se considera que A es la imagen y B es el elemento estructural. El elemento estructural es en morfología matemática lo que la máscara o núcleo de convolución es en los filtros lineales. Los elementos estructurales más comunes son los conjuntos que están 4-conectados(N_4), y 8-conectados(N_8), que se ilustran en las figuras 4.8.1 y 4.8.2:

0	1	0
1	1	1
0	1	0

Figura 4.8.1: Elemento estructural N_4

1	1	1
1	1	1
1	1	1

Figura 4.8.2: Elemento estructural N_8

La operación de dilatación hace que los objetos se expandan y la cantidad o la forma en que crezcan depende del elemento estructurante (*figura 4.8.3*).

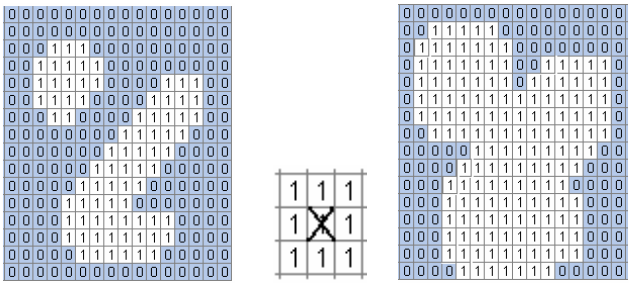


Figura 4.8.3. Dilatación: A) imagen original, B) elemento estructurante N8, C) imagen dilatada.

Erosión.- Combina dos conjuntos usando la resta vectorial y es el dual de la dilatación:

$$A \ominus B = \bigcup_{\beta \in B} (A - B) \qquad (4.8.2)$$

En otras palabras, son los puntos A para los cuales todas las posibles traslaciones definidas por B también están en A (*figuras 4.8.4 y 4.8.5*).

Figura 4.8.4. Erosión con un elemento estructurante de 1x2

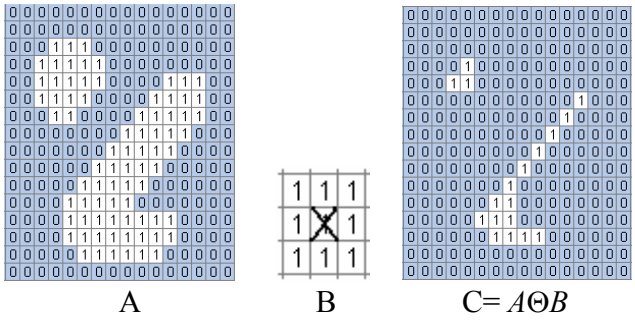
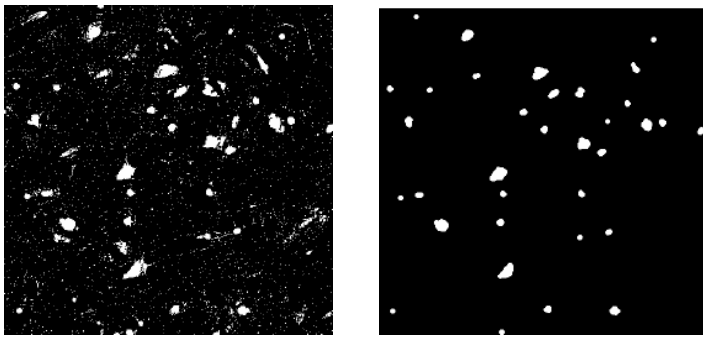


Figura 4.8.5. Erosión: A) imagen original, B) elemento estructurante de 3x3, C) imagen erosionada.

Apertura.- La apertura de A por B es una erosión de A por B seguida de la dilatación utilizando en ambas operaciones el mismo elemento estructurante.

$$A \circ B = (A \ominus B) \oplus B \qquad (4.8.3)$$

Se utiliza para alisar el contorno de un objeto, rompe uniones pequeñas uniones entre objetos y elimina componentes ruidosas (*puntos blancos*). En la figura 4.8.6 se utiliza la apertura como un filtro de tamaño, de esta manera se eliminan los objetos pequeños menores a un elemento estructurante y también elimina el ruido contenido en la imagen.



A B

Figura 4.8.6. Apertura: A) imagen original, B) imagen resultante después de aplicar la apertura.

Cerradura.- La cerradura de A por B es la dilatación de A por B seguida de la erosión utilizando en ambas operaciones el mismo elemento estructurante.

$$A \bullet B = (A \oplus B) \Theta B \qquad (4.8.4)$$

La Cerradura tiende a eliminar pequeños agujeros (*puntos negros*) del objeto, fusiona pequeños rompimientos y rellena pequeñas entradas al objeto como se muestra en la figura 4.8.7.

A B C

Figura 4.8.7. Cerradura: A) imagen en niveles de gris, B) imagen en blanco y negro C) imagen resultante después de aplicar cerradura.

Extracción de Frontera.- La frontera de un conjunto *A*, escrita como β(A), se puede obtener erosionando *A* por *B* y luego se calcula la diferencia entre *A* y su erosión. La figura 4.8.8 muestra el resultado de aplicar la extracción de frontera a una imagen binaria usando un elemento estructural de 3x3 estándar[1].

A) B)

Figura 4.8.8: Extracción de frontera: A) imagen binaria,
B) resultado de aplicar extracción de frontera.

[1] Imagen extraída de la Guía de referencia de MatLab versión 7.0

4.9. Extracción de contornos

La manera mas común de detectar contornos es aplicar algún tipo de derivada o diferencial, aplicado en un vecindario pequeño (*máscara*). La derivada nos permite calcular variaciones entre un punto y su vecindario. Viendo la imagen como una función, un contorno implica una discontinuidad en dicha función, es decir donde la función tiene un valor de gradiente o derivada alta (*ver figura 4.9.1*).

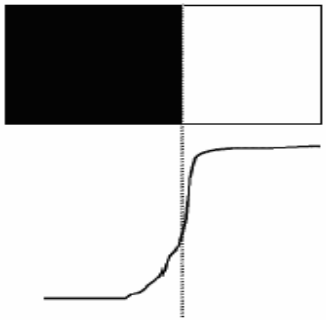


Figura 4.9.1: Gráfica de una imagen discontinua.

Operadores de gradiente

Las técnicas clásicas de detección de contornos se basan en encontrar la derivada respecto a los ejes x, y, o gradiente. El gradiente de una función se define como un vector bidimensional perpendicular al borde:

$$G[f(x,y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x} f(x,y) \\ \frac{\partial}{\partial y} f(x,y) \end{bmatrix} \qquad (4.9.1)$$

donde el vector G apunta en la dirección de variación máxima de f en el punto(x,y) por unidad de distancia con la magnitud dada por:

$$|G| = \sqrt{G_x^2 + G_y^2} \qquad (4.9.2)$$

En la práctica se aproxima la magnitud del gradiente con valores absolutos:

$$|G| \approx |G_x| + |G_y| \qquad (4.9.3)$$

Para calcular la derivada en la ecuación (4.9.1) se pueden utilizar las diferencias de primer orden entre dos píxeles adyacentes:

$$G_x = \frac{f(x+\nabla x) - f(x-\nabla x)}{2\nabla x} \qquad (4.9.4)$$

$$G_y = \frac{f(y+\nabla y) - f(y-\nabla y)}{2\nabla y} \qquad (4.9.5)$$

$$g(x,y) = \begin{cases} 1 & si \quad G[f(x,y)] > T \\ 0 & si \quad G[f(x,y)] \le T \end{cases} \qquad (4.9.6)$$

donde *T* es un valor de umbral no negativo entre 0 y 255 para imágenes en niveles de gris. Sólo los píxeles de borde cuyo gradiente excedan el valor de *T* se consideran importantes.

Operadores de Sobel

Los operadores de gradiente tienen la desventaja de aumentar el ruido en la imagen, tanto los operadores de *Sobel* como el resto de operadores de vecindad tienen la propiedad de suavizar la imagen, eliminando parte del ruido y minimiza la aparición de falsos contornos. A partir de las ecuaciones (4.9.4.) y (4.9.5.), las derivadas basadas en los operadores de *Sobel* son:

$$G_x = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \qquad (4.9.7)$$

$$G_y = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \qquad (4.9.8)$$

donde los valores de z en la región de la figura 4.9.2 (a), son los niveles de gris de los píxeles solapados por las máscaras en cualquier localización de la imagen. A partir de las ecuaciones (4.9.7) y (4.9.8) se determinan las máscaras para obtener G_x y G_y para el punto central (*figuras 4.9.2 a y b*). Una vez que se ha obtenido la magnitud del gradiente, se puede decidir si un determinado punto es de borde o no aplicando la ecuación (4.9.6) obteniendo como resultado una imagen binaria.

$$\begin{bmatrix} z_1 & z_2 & z_3 \\ z_4 & z_5 & z_6 \\ z_7 & z_8 & z_9 \end{bmatrix} \qquad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

(a) **(b)** **(c)**

Figura 4.9.2: (a) Región de la imagen de dimensión 3x3, (b) Máscara usada para obtener G_x en el punto central, (c) Mascara usada para obtener Gy en el mismo punto central.

Por ejemplo, supongamos que la matriz de la figura 4.9.3 representa una imagen y se desea calcular el gradiente en el píxel marcado con el punto negro. La región de la máscara en donde se desea calcular el gradiente está marcada con los puntos en blanco.

$$\begin{bmatrix} \circ 0 & \circ 8 & \circ 8 & 8 & 8 & 8 \\ \circ 1 & \bullet 8 & \circ 9 & 7 & 6 & 8 \\ \circ 2 & \circ 3 & \circ 4 & 8 & 8 & 8 \\ 2 & 2 & 2 & 8 & 8 & 8 \\ 2 & 2 & 2 & 2 & 8 & 8 \end{bmatrix}$$

Figura 4.9.3: Imagen muestra

Para calcular el gradiente aplicamos las ecuaciones (4.9.7) y (4.9.8), obteniendo $G_x = 30 - 4 = 26$ y $G_y = 12 - 24 = 12$. Utilizando la ecuación (4.9.3) se obtiene $|G| = 48$, si fijamos un valor de umbral *T=30*, el píxel marcado con el punto negro sería un punto de borde y si por el contrario, el umbral es mayor que 48, dicho punto no sería punto de borde. La figura 4.9.4 muestra un ejemplo de detección de bordes utilizando los operadores de Sobel.

A) **B)**

Figura 4.9.4: A) imagen original, B) imagen obtenida al aplicar la máscara de la figura 4.9.2

Operador Prewitt

El operador *Prewitt* es similar al de *Sobel*, lo único que cambia son los coeficientes de las máscaras, como se muestra en la figura 4.9.5.

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

(a) (b)

Figura 4.9.5: Máscaras de Prewitt. A) máscara usada para obtener Gx en el punto central, b) mascara usada para obtener Gy en el mismo punto.

La figura 4.9.6 es el resultado de aplicar los operadores de *Prewitt* a la imagen 4.9.4 (A)

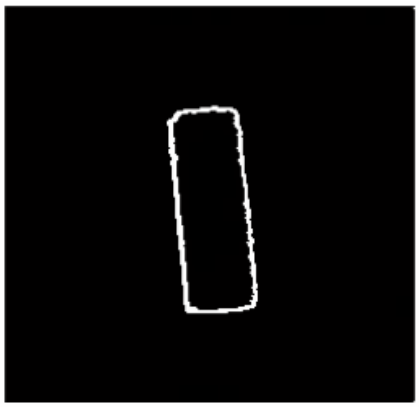


Figura 4.9.6: Operador de Prewitt.

Operador de Roberts

El operador de *Roberts* es muy simple y trabaja muy bien en imágenes binarias. Existen dos formulas para calcular el operador de *Roberts*:

$$\sqrt{[f(x,y)-f(x-1,y-1)]^2+[f(x,y-1)-f(x-1,y)]^2} \quad (4.9.9)$$

$$|f(x,y)-f(x-1,y-1)|+|f(x,y-1)-f(x-1,y)| \quad (4.9.10)$$

La más usada es la ecuación (4.9.10) por ser más simple y utilizar menos coste de cómputo. En las figura 4.9.7 se muestran la imagen obtenida por aplicación de la ecuación (6) a la imagen 4.9.4.(a) utilizando los operadores de *Roberts*.

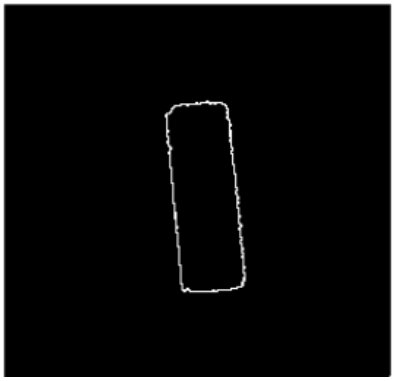


Figura 4.9.7: Operador de Roberts

Mascaras de Kirsch

Las máscaras de *Kirch* se denominan también de brújula porque se definen considerando una máscara simple y rotándola en las ocho direcciones de la brújula. Las máscaras se definen como sigue en la figura 4.9.8.

$$k_0 \equiv \begin{bmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{bmatrix} \quad k_1 \equiv \begin{bmatrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & -3 \end{bmatrix} \quad k_2 \equiv \begin{bmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix} \quad k_3 \equiv \begin{bmatrix} 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}$$

$$k_4 \equiv \begin{bmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{bmatrix} \quad k_5 \equiv \begin{bmatrix} -3 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & 5 & -3 \end{bmatrix} \quad k_6 \equiv \begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \end{bmatrix} \quad k_7 \equiv \begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \end{bmatrix}$$

Figura 4.9.8: Mascaras de Kirsch en las ocho direcciones.

Para cada punto de la imagen se obtienen 8 valores, resultantes de la convolución con cada una de las máscaras, el valor del módulo del gradiente es el máximo de esos ocho valores. En la figura 4.9.9 se muestra la imagen del módulo del gradiente obtenido con el operador *Kirch* a partir de la figura 4.9.4. (a).

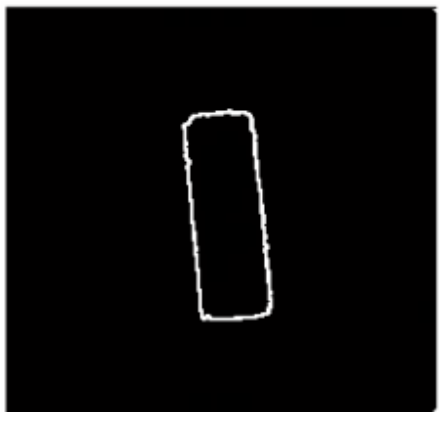


Figura 4.9.9: operador de Kirsch

Operador Laplaciano

El Laplaciano de una función 2D *f(x,y)* es un operador de segunda derivada definido como:

$$\nabla^2 f = \frac{d^2 f}{dx^2} + \frac{d^2 f}{dy^2} \qquad (4.9.11)$$

La ecuación anterior se puede implementar en forma digital como:

$$\nabla^2 f = 4z_5 - (z_2 + z_4 + z_6 + z_8) \qquad (4.9.12)$$

donde los coeficientes z se han definido en la figura 4.6.2 (a) y el requisito básico es que los coeficientes asociados con el píxel central y los coeficientes asociados con el resto de píxeles sean negativos. Puesto que el Laplaciano es una derivada, la suma de los coeficientes debe ser cero. Por lo tanto, la respuesta es cero siempre que el punto en cuestión y sus vecinos tienen el mismo valor.

Las tres máscaras Laplacianas de la figura 4.9.10 representan diferentes aproximaciones del operador Laplaciano y son capaces de detectar bordes en todas las direcciones espaciales. En la figura 4.9.11 se muestran los resultado de aplicar los operadores de la figura 4.6.10.

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

(a) **(b)** **(c)**

Figura 4.9.10: Máscaras de operadores Laplacianos

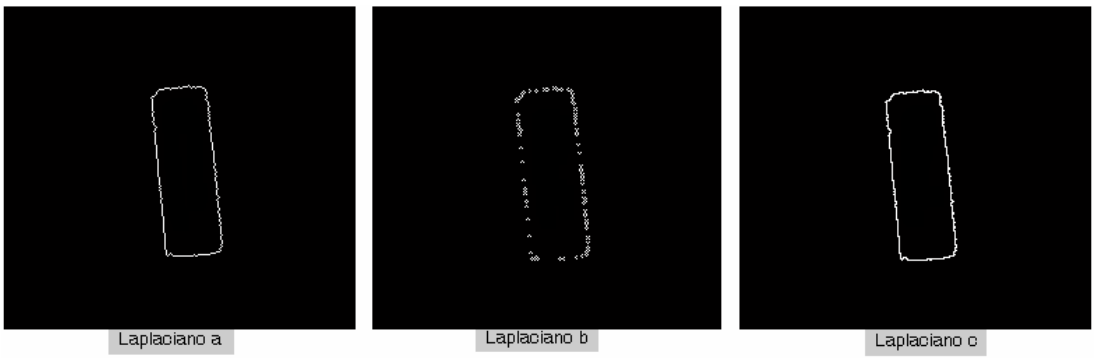


Figura 4.9.11: Moperadores Laplaciano

Capítulo 5

SEGMENTACIÓN

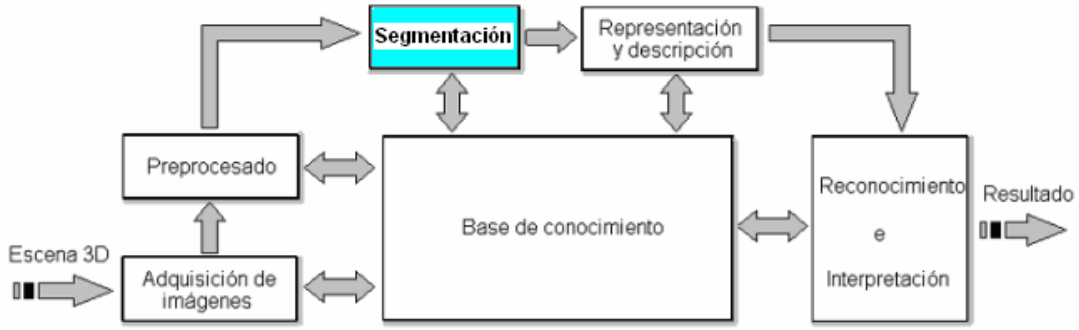


Figura 5.1.0: Etapas fundamentales en el reconocimiento automático de objetivos

En la figura 5.1.0 se resalta la etapa de segmentación que es un proceso por el cual se particiona o divide la imagen en sus componentes (*objetos de interés*) para su uso posterior. La segmentación autónoma es una de las tareas más difíciles del procesamiento de imágenes ya que esta etapa determina el éxito o el fracaso del proyecto.

La segmentación puede ser orientada en regiones (*área de la imagen en la que sus píxeles poseen propiedades similares de intensidad o de color*) o a bordes (*líneas que separan dos regiones*). Tanto la detección de regiones como la de bordes implican una manipulación de la imagen original, donde los valores de los píxeles originales son modificados mediante ciertas operaciones de transformación u operadores.

5.1 Detección de regiones basada en umbrales

Si analizamos el histograma de una imagen en niveles de gris que corresponde a una imagen con un objeto claro sobre fondo oscuro (*ver figura 5.1.1*), de tal forma que los píxeles del objeto y del fondo tienen los niveles de gris agrupados en dos modos dominantes, como podemos ver en el histograma de la figura 5.1.2, una forma de extraer los objetos del fondo es elegir un umbral "*T*" que separe dichos grupos. Cualquier punto*(x, y)* para el que *f(x, y)* > *T* se asigna al objeto, en caso contrario al fondo; en otras palabras, se pueden separar los píxeles en dos regiones si asignamos con un valor de intensidad cero a todos los píxeles cercanos o iguales a cero (*fondo negro*) y con 255 al resto de los valores (*objeto*).

Figura 5.1.1: Imagen en niveles de gris

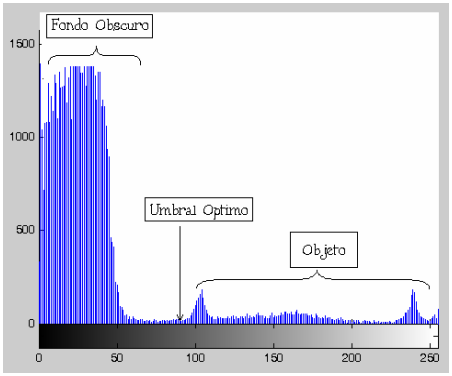


Figura 5.1.2: Histograma

Analizando el histograma, se deduce experimentalmente que un umbral óptimo que segmenta a la imagen en dos regiones es = 97. La primera región correspondiente al fondo y la otra al objeto. Al utilizar un umbral mas pequeño, los puntos claros correspondientes al fondo se unen con el objeto a segmentar: por el contrario, si tomamos un umbral mas grande corremos el riesgo de tomar zonas del objeto que son un poco obscuras y unirse con el fondo (*ver figuras 5.1.3, 5.1.4 y 5.1.5*).

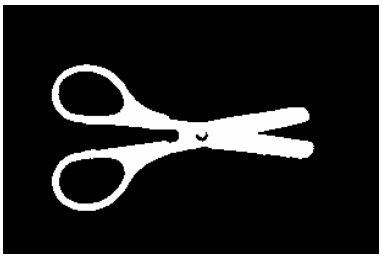


Figura 5.1.3: umbral óptimo 97

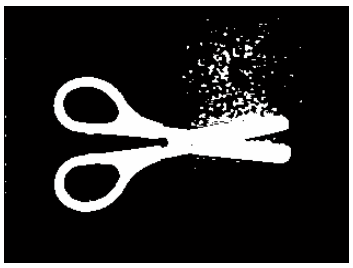


Figura 5.1.4: umbral = 45

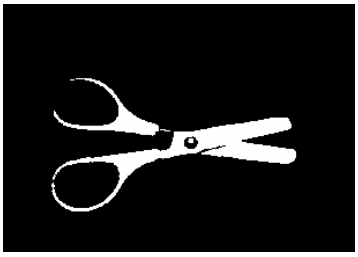


Figura 5.1.5: umbral =140

5.2 Método de segmentación de Otsu

La importancia del método de Otsu radica en que es automático, es decir, no necesita supervisión humana ni información previa de la imagen antes de su procesamiento. Este método se emplea cuando hay una clara diferencia entre los objetos a extraer respecto del fondo de la escena. En este método, se elige el umbral óptimo maximizando la varianza entre clases mediante una búsqueda exhaustiva. A medida que el número de clases de una imagen aumenta, el método de Otsu necesita mucho más tiempo para seleccionar un umbral multinivel adecuado.

Al aplicar un umbral *T*, la imagen en escala de grises, *f(x,y)*, quedará binarizada; etiquetando con "*1*" los píxeles correspondientes al objeto y con "*0*" aquellos que son del fondo. Si los objetos son claros respecto del fondo, se aplica la siguiente fórmula:

$$g(x,y) = \begin{cases} 1 \Leftrightarrow f(x,y) > T \\ 0 \Leftrightarrow f(x,y) \leq T \end{cases} \qquad (5.2.1)$$

Descripción del Método de Otsu para un umbral óptimo

El número de píxeles con nivel de gris *"i"* se denota como "f_i", y la probabilidad de ocurrencia del nivel de gris *"i"* en la imagen está dada por:

$$P_i = \frac{f_i}{N * M} \qquad (5.2.2)$$

Para la umbralización en dos niveles de una imagen, los píxeles son divididos en dos clases: **C1**, con niveles de gris [*1,..., t*]; y **C2**, con niveles de gris [*t+1,..., L*]. Entonces, la distribución de probabilidad de los niveles de gris para las dos clases son:

$$C_1 = \frac{p_1}{\omega_1(t)},\ldots,\frac{p_t}{\omega_1(t)} \tag{5.2.3}$$

$$C_2 = \frac{p_{t+1}}{\omega_2(t)},\frac{p_{t+2}}{\omega_2(t)},\ldots,\frac{p_L}{\omega_2(t)} \tag{5.2.4}$$

donde:

$$\omega_1(t) = \sum_{i=1}^{t} p_i \tag{5.2.5}$$

y

$$\omega_2(t) = \sum_{i=t+1}^{L} p_i \tag{5.2.6}$$

y la media para la clase C1 y para la clase C2 son:

$$\mu_1 = \sum_{i=1}^{t} \frac{i * p_i}{\omega_1(t)} \tag{5.2.7}$$

y

$$\mu_2 = \sum_{i=t+1}^{L} \frac{i * p_i}{\omega_2(t)} \tag{5.2.8}$$

Otsu definió la variancia entre clases de una imagen umbralizada como:

$$\sigma_B^2 = \varpi_1(\mu_1 - \mu_T)^2 + \varpi_2(\mu_2 - \mu_T)^2 \tag{5.2.9}$$

también verificó que el umbral óptimo ***t**** se elige de manera que σ_B^2 sea máxima:

$$t^* = \underset{t}{Max}\left\{\sigma_B^2(t)\right\} \quad 1 \le t \le L \tag{5.2.10}$$

Ejemplo:

Supongamos un imagen de N*M=100 píxeles con cuatro niveles de gris comprendidos en [1,4] (*1 el negro, 4 el blanco*) y supongamos también que el número de píxeles con nivel de gris es el siguiente:

Nivel	# de píxeles f_i
1	10
2	20
3	30
4	40
Total	**100**

La Probabilidad de ocurrencia $P_i = \frac{f_i}{N * M}$ para cada nivel sería:

Nivel	# de píxeles f_i	Probabilidad de ocurrencia p_i
1	10	P_1=10/100=**0.1**
2	20	P_2=20/100=**0.2**
3	30	P_3=30/100=**0.3**
4	40	P_4=40/100=**0.4**

para una umbralización en dos niveles de esta imagen tomemos t=2 de manera que la clase C1 consista en los niveles de gris 1 y 2, y la clase C2 contenga los niveles 3 y 4.

por lo tanto, la Distribución de Probabilidad para

$$\omega_1(t) = \sum_{i=1}^{t} p_i \quad \text{y} \quad \omega_2(t) = \sum_{i=t+1}^{L} p_i$$ es:

Nivel	# de píxeles f_i	Probabilidad de ocurrencia p_i	Distribución de Probabilidad
1	10	P_1=10/100=**0.1**	$\omega_1(t)$= *0.1+0.2=**0.3***
2	20	P_2=20/100=**0.2**	
3	30	P_3=30/100=**0.3**	$\omega_2(t))$=*0.3+0.4=**0.7***
4	40	P_4=40/100=**0.4**	

Se comprueba que $\omega_1(t)+\omega_2(t)=1$.

Por último, la media para la clase C_1 y para la clase C_2 estará dada por:

$$\mu_1 = \sum_{i=1}^{2} \frac{i * p_i}{\omega_1(t)} = \frac{1*0.1 + 2*0.2}{0.3} \approx 1.667$$

$$\mu_2 = \sum_{i=3}^{4} \frac{i * p_i}{\omega_2(t)} = \frac{3*0.3 + 4*0.4}{0.7} \approx 3.57$$

y

$$\mu_t = \varpi_1 \mu_1 + \varpi_2 \mu_2 = 0.3*1.667 + 0.7*3.57 \approx 3$$

$$\sigma_B^2 = 0.3(1.667 - 3)^2 + 0.7(3.57 - 3)^2 = 0.53 + 0.227 = 0.75$$

El valor óptimo puede encontrarse al buscar en el rango $1 \le t \le L$ el valor de variancia entre clases que dé máxima.

Capítulo 6

REPRESENTACIÓN Y DESCRIPCIÓN

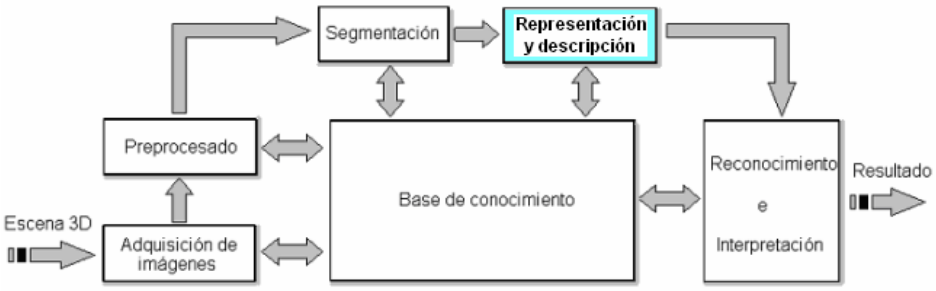


Figura 6.1.1: Etapa de representación y descripción del objeto.

Una vez que los objetos de interés han sido aislados, el siguiente paso para el reconocimiento del objeto es: La representación y descripción (*ver figura 6.1.1*). En esta etapa, a la imagen le son extraídos un conjunto de características que describen las propiedades físicas de los objetos como son: el color, textura, área, perímetro, ancho, alto, promedio de intensidad, centroide, orientación, color, número de hoyos, entre otros.

Las propiedades que deben tener estos descriptores de características son:

- Ser capaces de hacer una adecuada discriminación: deben proporcionar valores numéricos diferentes para objetos de clases distintas.
- Ser suficientemente fiables: debe de haber cambios numéricos pequeños para objetos de la misma clase.
- Rapidez: El número de descriptores debe ser pequeño y calcularse en un tiempo aceptable.

6.1. Descriptor de Color

El color es una característica importante para la clasificación y descripción de objetos en la visión por computadora. Existen diferentes modelos de color los cuales se describen a continuación:

Modelo RGB

El modelo RGB(*Red, Green, Blue*), maneja 3 canales, uno para el rojo, uno para el verde y uno para el color azul, la imagen resultante tiene tonalidades que surgen de las combinaciones entre los colores primarios. En esta codificación, por cada píxel se necesitan 24 bits, es decir 3 bytes, codificando cada color primario con 1 byte(*8 bits*) que pueden tomar 256 valores con un rango de 0 a 255. Los píxeles RGB son tridimensionales, $pixel_{RGB} = f(R,G,B)$, donde el valor de *f* es la intensidad del color rojo, verde y azul en las coordenadas *(x,y)* del $pixel_{RGB}$. De esta forma los colores se representan en coordenadas cartesianas dentro de un cubo unitario [13] (*figura 6.1.2*).

Cada color se representa como un vector del origen y la diagonal principal corresponde a la escala de grises. En este modelo se basan las cámaras y receptores de televisión. Sin embargo, no es recomendable aplicarlo al procesamiento de imágenes y visión.

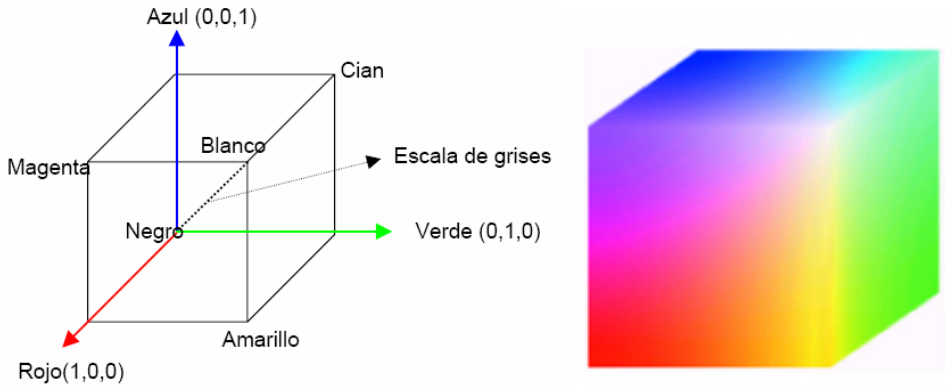


Figura 6.1.2: Distribución de colores en el cubo RGB

Modelo YCbCr

En el modelo YCbCr el color es representado por la luminancia (Y) y por dos valores diferentes de color (Cb y Cr) que son características colorimétricas del color. La luminancia es la cantidad lineal de luz que puede ser calculada como la suma ponderada de los componentes lineales del espacio de color RGB [13], (ver f*igura 6.1.3)*.

La obtención de este espacio de color a partir del RGB es la siguiente:

$$\begin{aligned} Y &= 0.2989 * R + 0.5866 * G + 0.1145 * B \\ C_b &= -0.1688 * R - 0.3312 * G + 0.5 * B \\ C_r &= 0.5 * R - 0.4184 * G - 0.08116 * B \end{aligned} \qquad (6.1.1)$$

siendo R, G y B son los valores del canal rojo, verde y azul respectivamente .

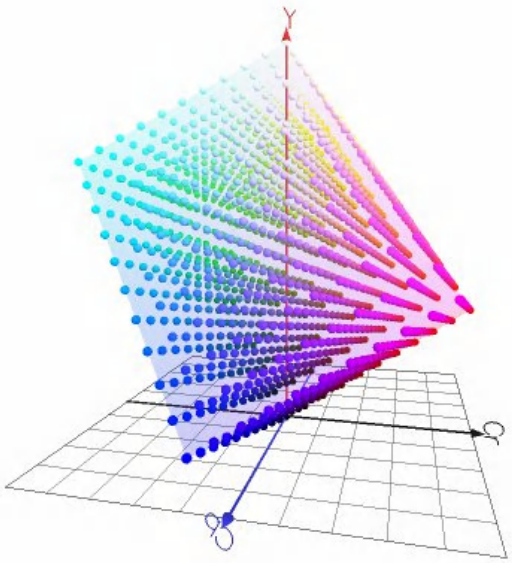


Figura 6.1.3: Distribución de color YCbCr

Modelo XYZ

Es el primer espacio de color estandarizado en 1931. Estas componentes no son reales, sino imaginarias, pero cualquier color se puede definir como combinación de ellas. De esta forma, toda la información de intensidad esta comprendida únicamente en la componente Y.

Estos valores se pueden obtener de las componentes RGB como:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.607 & 0.174 & 0.200 \\ 0.299 & 0.587 & 0.114 \\ 0.000 & 0.066 & 1.116 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \qquad (6.1.2)$$

En la figura 6.1.4. se muestra cómo los colores se distribuyen en la curva en función de su longitud de onda. Se puede observar que con este espacio de color no es fácil la separación de colores [15].

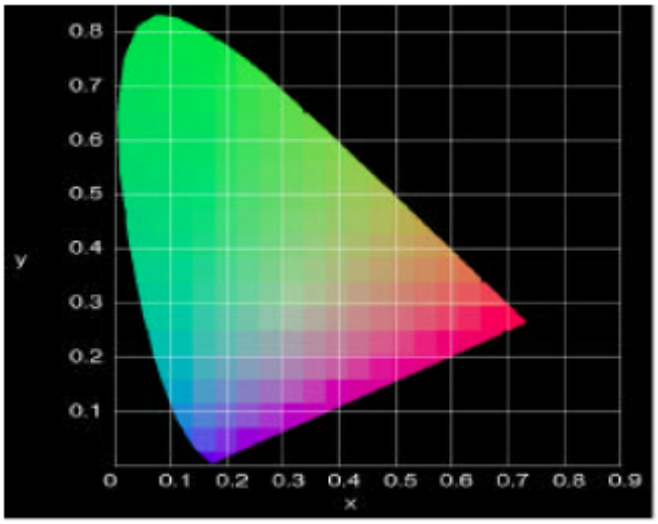


Figura 6.1.4: Distribución del color en el modelo XYZ

Modelo CIE(Lab)

Una vez que se dispone de las coordenadas XYZ, puede optarse por trabajar en distintos espacios CIE. Entre ellos, el CIE(*Lab*) es muy popular por tratarse de un espacio uniforme y adecuado para la segmentación. La mayor ventaja de estos sistemas es que el parecido entre dos colores se puede medir por distancia euclideana [15], (*figura 6.1.5*).

Para obtener los valores Lab de una imagen color, pueden usarse las ecuaciones:

$$L = 25\left(100\left(\frac{Y}{Y_0}\right)^{1/3}\right) - 16 \qquad (6.1.3)$$

$$a = 500\left[\left(\frac{X}{X_0}\right)^{1/3} - \left(\frac{Y}{Y_0}\right)^{1/3}\right] \qquad (6.1.4)$$

$$b = 200\left[\left(\frac{Y}{Y_0}\right)^{1/3} - \left(\frac{Z}{Z_0}\right)^{1/3}\right] \qquad (6.1.5)$$

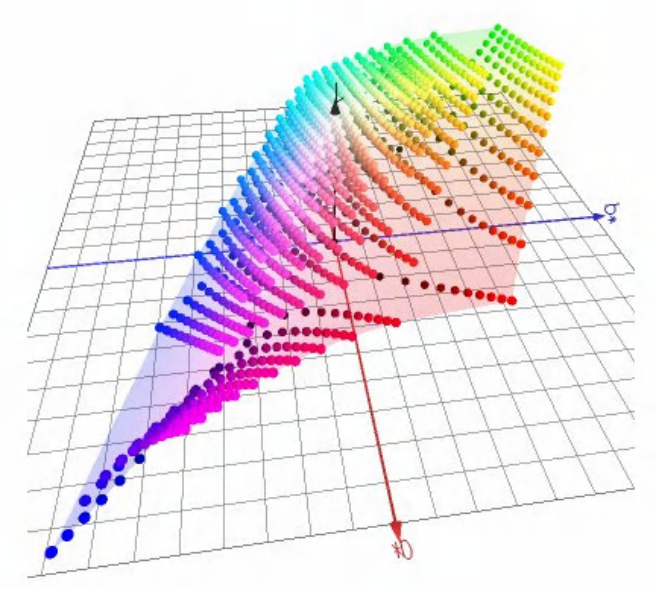


Figura 6.1.5: Espacio de color CIELAB

Modelo HSI

En este modelo los colores se representan mediante sus propiedades: Tono, Saturación e Intensidad (*HSI: Hue, Saturation, Intensity*), que describen color, pureza del color y brillo. El tono equivale al color que nosotros apreciamos. La saturación se refiere a la pureza de dicho color (su grado de mezcla con los otros colores primarios. Por último, la intensidad puede identificarse como el brillo de la imagen.

Un valor de color es representado como un punto en el cilindro. El tono es el ángulo de rotación sobre la superficie circular del cilindro. El rojo es definido a cero grados. La saturación es representada por la longitud del vector desde el centro del círculo al valor del color en cuestión. Para los colores puros, este vector se extiende al borde de la superficie cilíndrica. La intensidad es representada por la longitud del vector desde la base del cilindro al valor del color [15], (*ver figura 6.1.6*).

La conversión de la representación *RGB* a *HSI* puede ser llevada a cabo considerando las siguientes ecuaciones:

$$\theta = \cos^{-1}\left(\frac{1/2((R-G)+(R-B))}{(R-G)^2+(R-B)(G-B)^{1/2}}\right) \qquad (6.1.6)$$

$$H = \begin{cases} \theta, & si\ G \geq B \\ 2\pi - \theta & en\,caso\,contrario \end{cases} \qquad (6.1.7)$$

$$S = 1 - \frac{\min(R,G,B)}{I} \qquad (6.1.8)$$

$$I = \frac{R+G+B}{3} \qquad (6.1.9)$$

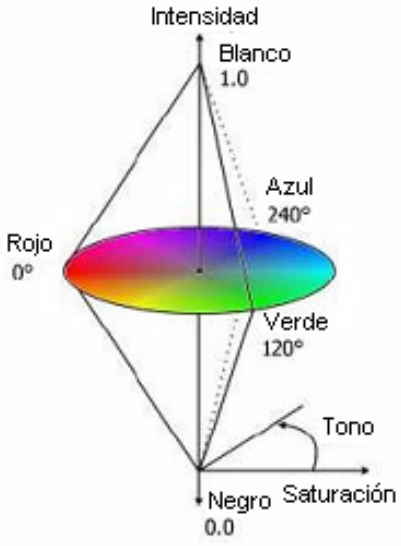


Figura 6.1.6 Espacio de color HSI

6.2. Textura

La textura es otro descriptor importante y muy utilizado en el reconocimiento, nos describe qué tan homogénea será la imagen. Los descriptores de textura se basan siempre en una vecindad, ya que la textura se define para regiones y no para píxeles individuales. Es difícil encontrar un solo descriptor de la textura, ya que existen varios problemas asociados a ellos: El detector perfecto debería ser insensible a rotaciones y a escalamientos.

Matriz de coocurrencia

En el análisis de texturas, la extracción de características se realiza a partir de la distribución estadística con la que se observan combinaciones de determinadas intensidades en posiciones relativas de la imagen. La matriz de coocurrencia es una matriz cuadrada en la que el número de filas y columnas coincide con el número de niveles de gris (G) en la imagen a analizar y donde cada elemento de la matriz *C(i,j)* contiene la frecuencia relativa con la que dos píxeles de la imagen I, con intensidades *i* y *j* respectivamente y separados por una distancia *d* y un ángulo θ, ocurren en una determinada vecindad.

Dicho de otro modo, el elemento $C(i,j|d,\theta)$ contiene la probabilidad de que dos píxeles cualesquiera a una distancia *d* y un ángulo θ, tengan respectivamente niveles de gris *i* y *j*.

La implementación matemática de la matriz de coocurrencia es la siguiente:

$$C_{ij} = \left|\left\{\left((r,s),(t,v)\right) : I(r,s) = i, I(t,v) = j\right\}\right| \qquad (6.2.1)$$

donde $(r,s),(t,v) \in NxN, (t,v) = (r+dx, s+dy)$

Ejemplo:

Tomando como base la matriz de la figura (6.2.1.), con una distancia de un píxel "*d=1*" y direcciones a 0°, determinemos la matriz de coocurrencia.

Como esta imagen únicamente contiene tres niveles de gris (0, 1 y 2), se crea una matriz de 3x3 para cada rotación. El cálculo de la matriz de coocurrencia para cada dirección se muestra en las figuras 6.2.2, 6.2.3 y 6.2.4.

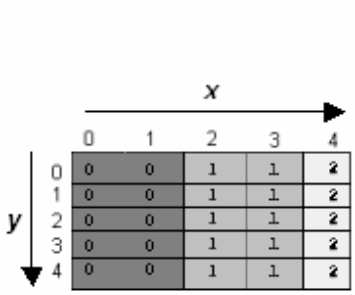


Figura 6.2.1: imagen con tres niveles de gris.

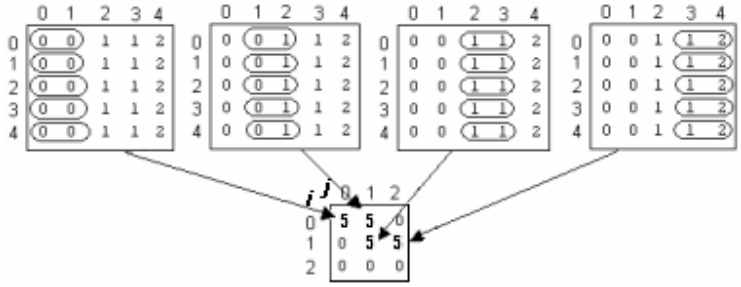


Figura 6.2.2: Matriz de coocurrencia para d=1 a 0°

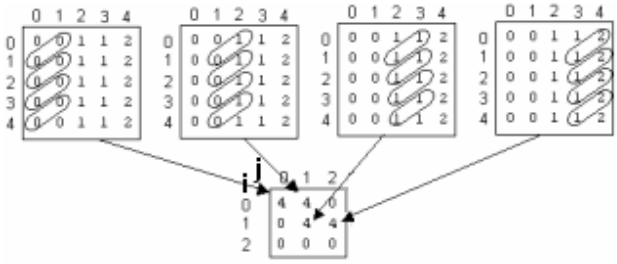


Figura 6.2.3: Matriz de coocurrencia para d=1 a 45°.

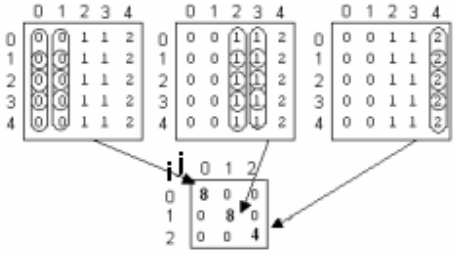


Figura 6.2.4: Matriz de coocurrencia para d=1 a 90 °.

Así, cuanto mayores sean los valores de la diagonal principal de la matriz de coocurrencia, más homogénea será la textura que representa, mientras que cuanto más repartidos estén los valores fuera de la diagonal más heterogénea será.

A continuación se presentan algunos métodos de detección de rasgos o características de textura que se calculan a partir de la matriz de coocurrencia (*en nuestro caso se utilizó el promedio de las tres matrices de coocurrencia para 0°, 45° y 90°*).

Energía: El descriptor de energía de una textura extraído de la matriz de coocurrencia está dado por la siguiente ecuación:

$$Energía = \sum_i \sum_j C_{ij}^2 \qquad (6.2.2)$$

donde, C_{ij} es cada elemento de la matriz de coocurrencia y varían desde 0 hasta el número total de niveles de gris.

De la ecuación (6.2.2) se puede decir que cuando todos los valores de *cij* son semejantes, (*mayor dispersión en la diagonal principal de la matriz*), el valor de la energía será menor, por el contrario si ocurre que en la diagonal principal se dan mayores picos de intensidad el descriptor será mayor. La propiedad de energía da una idea de la suavidad de la textura, y esto se refleja en la ubicación de sus probabilidades en la matriz de coocurrencia.

Entropía: Es una medida de la aleatoriedad contenida en la matriz de coocurrencia. Se obtiene con la siguiente fórmula:

$$Entropía = -\sum_i \sum_j C_{ij} \log_2 \left| C_{ij} \right| \qquad (6.2.3)$$

A medida que todos los elementos de la matriz son similares, este descriptor aumenta su valor, siendo máximo en el caso que todos los elementos de la matriz fueran iguales.

Contraste: El contraste de una textura proporciona información acerca de las variaciones bruscas de color en la imagen y está dado por la siguiente expresión:

$$Contraste = \sum_i \sum_j \left| i - j \right|^2 C_{ij} \qquad (6.2.4)$$

Como se observa en la ecuación (6.2.4), el valor del contraste aumentará, si existen más elementos de la matriz de coocurrencia alejados de la diagonal principal. En una textura de características suaves y uniformes su contraste será bajo, mientras que si presenta un aspecto rugoso o irregular su contraste presentará un valor alto.

Homogeneidad Local: El descriptor de homogeneidad local proporciona información sobre la regularidad local de la textura. La descripción matemática de este descriptor esta dada por la siguiente ecuación.

$$Homogeneidad = \sum_i \sum_j \frac{C_{ij}}{1+(i-j)^2} \qquad (6.2.5)$$

Este descriptor aumentará cuando la distancia *i-j* sea mínima. Lo cual indica que mientras los elementos de la matriz de coocurrencia estén más próximos a la diagonal principal mayor será el valor de la homogeneidad local.

6.3. Rasgos geométricos de un objeto

Son mediciones que pueden ser usados para describir un objeto, se obtienen a partir de los píxeles que componen la región del objeto o los píxeles obtenidos del contorno del objeto.

Área.- El área de un objeto en una imagen binaria es igual a la suma de todos los puntos correspondientes al objeto. Se realiza un recorrido por todas las columnas C y los renglones R de la imagen y se van sumando los píxeles correspondientes al objeto:

$$A = \sum_{x=0}^{C} \sum_{y=0}^{R} f(x,y) \qquad (6.3.1)$$

Perímetro.- La fórmula es semejante a la del área, solamente se suman los píxeles del contorno del objeto. Un píxel es de contorno (*vecindad N_4*), si tiene al menos un píxel de fondo a la izquierda, derecha, arriba o abajo.

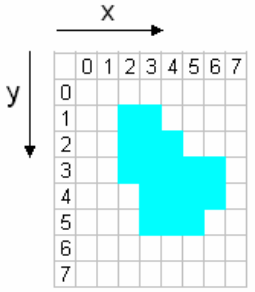
$$P = \sum_{x=0}^{C} \sum_{y=0}^{R} f(x,y) \quad \forall f(x,y) \text{ con algún píxel de } N_4(f(x,y)) = 0 \qquad (6.3.2)$$

Centroide.- Es un punto geométrico que se refiere a la parte central de la imagen, las coordenadas de este punto se calculan con las siguientes fórmulas:

$$\bar{x} = \frac{\sum_{x=0}^{C}\sum_{y=0}^{R} x * f(x,y)}{A} \qquad (6.3.3)$$

$$\bar{y} = \frac{\sum_{x=0}^{C}\sum_{y=0}^{R} y * f(x,y)}{A} \qquad (6.3.4)$$

Por ejemplo: Se desea calcular el área, perímetro y centroide del siguiente objeto



Aplicando la fórmula 6.3.1 obtenemos el cálculo del área:

$$Área = \sum_{x=0}^{C}\sum_{y=0}^{R} f(x,y) = 3+5+4+3+2 = 17$$

Para calcular el perímetro en la imagen se determinan los píxeles de frontera con conexión N_4:

$$Perímetro = 12 \quad \forall f(x,y) \text{ con algún píxel de } N_4(f(x,y)) = 0$$

Si aplicamos las fórmulas 6.3.3 y 6.3.4 para aplicar el centroide tenemos:

$$\bar{x} = \frac{\sum_{x=0}^{C}\sum_{y=0}^{R} x * f(x,y)}{A} = \frac{6+15+16+15+12}{17} = \frac{64}{17} = 3.76$$

$$\bar{y} = \frac{\sum_{x=0}^{C}\sum_{y=0}^{R} y * f(x,y)}{A} = \frac{6+15+14+12+7}{17} = \frac{54}{17} = 3.18$$

Redondeando los valores obtenidos al entero más próximo se determina las coordenadas del centroide en: C(4,3).

Factor de Compacidad (FC).- Este rasgo es de gran importancia y muy empleado para la clasificación. Se define como el perímetro al cuadrado del objeto dividido por 4π veces su área (*ecuación 6.4.5.*) de esta forma se compara la forma del objeto con un círculo.

$$FC = \frac{P^2}{4\pi(A)} \qquad (6.4.5)$$

En la tabla (6.3.1) se puede observar que el valor del Factor de Compacidad será = 1 para objetos circulares, un poco mayor a "1" para objetos cuadrados y se obtendrán valores mas grandes para otro tipo de objetos como el rectángulo.

La importancia de este rasgo característico radica en que es invariante a traslaciones, rotaciones y cambios de escala.

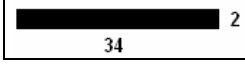
Figura	Medidas	Factor de Compacidad
Circulo 	Circunferencia= $2\pi r$ =6.28 Área = πr^2	$FC = \frac{(6.28)^2}{4(3.1416)(3.1416)} = \frac{39.44}{39.45} \approx 1$
Cuadrado 	Perímetro = 20 Área = 25	$FC = \frac{(20)^2}{4(3.1416)(25)} = \frac{400}{314.16} = 1.27$
Rectángulo 	Perímetro = 72 Área = 68	$FC = \frac{72^2}{4(3.1416)(68)} = \frac{5184}{854.48} = 6.06$

Tabla 6.3.1: Cálculo del Factor de Compacidad para diferentes figuras.

Elipse.- Lugar geométrico de los puntos de un plano cuya suma de distancias a dos puntos fijos del plano, F y F', es constante.

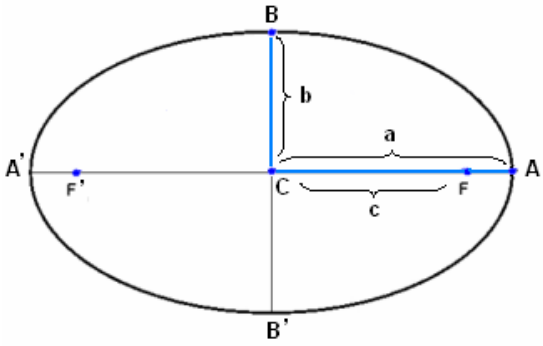


Figura 6.3.1. Elementos de una elipse: Semieje mayor $\overline{CA} = a$, semieje menor $\overline{CB} = b$ y la distancia focal $\overline{CF} = c$.

Excentricidad de una elipse.- Se define como el cociente entre la distancia focal "*c*" y el semieje mayor "*a*", su fórmula es:

$$e = \frac{c}{a} \qquad (6.4.6)$$

de la ecuación (6.4.6) se pueden observar lo siguiente:

- $a \geq c$ y por lo tanto el valor de e oscila entre 0 - 1.
- Cuando la e = 0, tenemos el caso de a = b y por lo tanto es un circunferencia (*figura 6.3.2*).
- Si e = 1, semieje mayor coincide con la distancia focal y el semieje menor es igual a 0. Será el caso de un segmento de recta (*figura 6.3.3*).

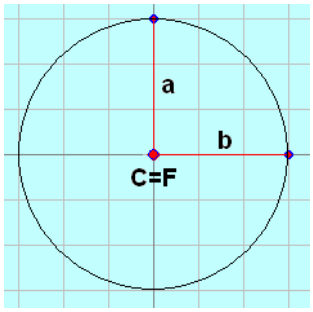


Figura 6.3.2: Circunferencia a=b y C=F.

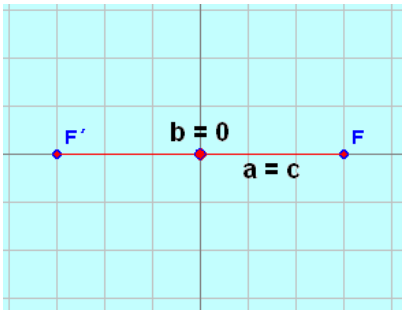


Figura 6.3.3: Segmento de Recta a=c y b=0.

En matlab se utiliza la función "regionprops" para determinar la excentricidad de un objeto de una imagen binaria.

Capítulo 7

RECONOCIMIENTO E INTERPRETACIÓN Y CÁLCULO DE LA DISTANCIA DEL OBJETO

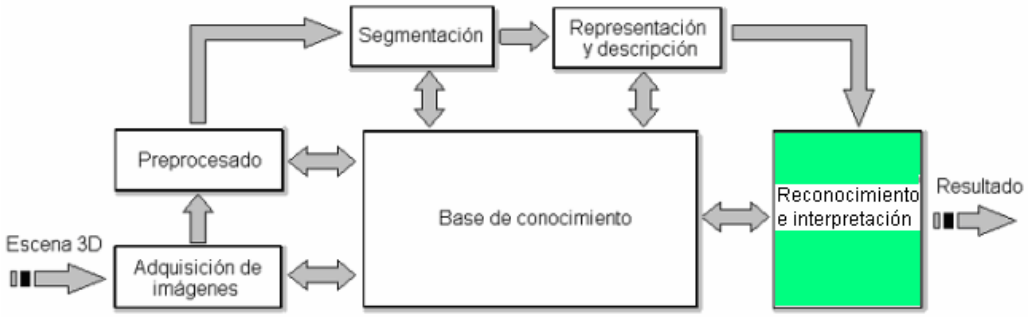


Figura 7.1.1: Etapa de reconocimiento e interpretación de características.

Como se muestra en la figura (7.1.1.), el último paso para el reconocimiento automático del objeto es la interpretación de los datos (*descriptores*) representados por características, para ello en el presente proyecto se implementó el clasificador *"K vecinos más próximos"*.

Otro de los puntos a tratar en este capítulo es el cálculo de la distancia a la que se encuentra el objeto con respecto a la cámara. Para ello se utiliza un láser orientado a un ángulo previamente determinado con respecto de la cámara. La matemática es muy simple y esta técnica trabaja muy bien para aplicaciones de visión de computadora que necesitan correr rápidamente.

7.1. Clasificador "K Vecinos más Próximos"

La clasificación por los *"K"* vecinos más próximos (*K Nearest Neighbor* o K-NN) es ampliamente utilizada en el reconocimiento de formas. Dado un vector a clasificar (*rasgos característicos del objeto a clasificar*) y un conjunto de vectores prototipo asignados a las diversas clases existentes (*base de conocimiento*). La regla consiste en calcular la distancia del primero a cada uno de los segundos, seleccionar los *"K"* vecinos más próximos y decidir por la clase más votada entre los mismos.

Principio teórico:

Sea $\vec{x}$ un vector de dimensión "*n*" a clasificar, sea M una base de datos de referencia construida de *N* vectores de dimensión "*n*" y se conoce la clase C_i a la cuál pertenecen los vectores de la base de referencia M. El clasificador K vecinos más próximos está basado en la estimación local de la densidad de probabilidad de la muestra $\vec{x}$ a partir de los "*K próximos vecinos*" de la base de referencia. Ver figura 7.1.2.

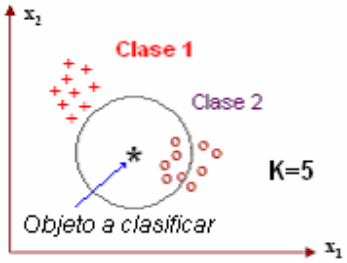


Figura 7.1.2: Clasificador KNN.

Sea $p(\vec{x}/C_i)$ la densidad de probabilidad. A partir de esta estimación, la regla de BAYES nos permite expresarlo en términos de la probabilidad a posteriori que la muestra $\vec{x}$ pertenezca a la clase C_i, tal que:

$$p_r(C_i/\vec{x}) = \frac{p(\vec{x}/C_i) * p_r(C_i)}{p(\vec{x})} = \frac{p_r(\vec{x}/C_i) * p_r(C_i)}{\sum_{k=1}^{c} p_r(\vec{x}/C_k) * p_r(C_k)} \qquad (7.1.1)$$

donde:

$p_r(C_i)$ = probabilidad de aparición de la clase C_i.

$p_r(\vec{x})$ = probabilidad de que la muestra $\vec{x}$ pertenezca a la clase C_i.

$p_r(C_i/\vec{x})$ = densidad de probabilidad condicional de la muestra $\vec{x}$ conociendo la clase C_i.

Partiendo de la base de referencia M (*base de aprendizaje*), se estiman las densidades de probabilidad $p(\vec{x}/C_i)$ para todas las clases C_i siguiendo dos métodos diferentes, produciendo dos reglas de decisión o afectación diferentes. El principio está basado sobre la búsqueda de "*K próximos vecinos*" de $\vec{x}$ sin importar la clase (*modo reagrupamiento general*) o en una clase C_i (*modo reagrupamiento por clase*).

Modo reagrupamiento General:

Sea "*V*" el volumen hiperesférico definido por la distancia "D" entre la muestra $\vec{x}$ y el K-ésimo vecino, la densidad de probabilidad conjunta $p(\vec{x}/C_i)$ es definida como $K_i/(N*V)$, siendo K_i el número de muestras que pertenecen a la clase C_i entre los *K* vecinos, normalizado con respecto al número total de muestras y dividido por el volumen que engloban los K-vecinos.

Si se hace la hipótesis que las probabilidades de aparición de cada clase son equiprobables $\forall i,j\ P_r(C_i)=P_r(C_j)$, entonces la ecuación (7.1.1) se convierte en:

$$p_r(C_i/\vec{x})=\frac{\frac{K_i}{N*V}}{\sum_{j=1}^{c}\frac{K_j}{N*V}}=\frac{K_i}{\sum_{j=1}^{c}K_j}=\frac{K_i}{K} \qquad (7.1.2)$$

donde:

C = Número total de clases

K = Número de "k" vecinos buscados

La clase a la cual pertenece la muestra $\vec{x}$ es determinada considerando el más grande número de prototipos pertenecientes a la clase C_i (*entre los k prototipos*). Es decir, que $\vec{x}$ es asociado a la clase mayoritariamente representada de entre los *K* próximos vecinos. Generalmente, el valor de *K* debe ser impar para evitar ambigüedades de clases que tienen el mismo numero *K* de vecinos. En el caso a 2 clases C_1 y C_2 : si $k_1/k_2>1$, entonces la clase ganadora será C_1 en el caso contrario la muestra $\vec{x}$ será asignada a la clase C_2 (*ver figura 7.1.3*).

Modo de reagrupamiento por clase

La densidad de probabilidad conjunta $p(\vec{x}/C_i)$ esta ahora definida como $K/(N*V_i)$. El número K de prototipos pertenecientes a la clase C_i son normalizados con respecto al total de prototipos (N) y divididos por el volumen V_i generado a partir de la k-ésima distancia. Sí se realiza la misma hipótesis que las probabilidades a priori de cada clase C_i son equiprobables $\forall i,j\ P_r(C_i)=P_r(C_j)$, entonces la ecuación (7.1.1) se convierte en:

$$p_r(C_i/\vec{x})=\frac{\frac{K_i}{N*V_i}}{\sum_{j=1}^{c}\frac{K_j}{N*V_i}}=\frac{1/V_i}{\sum_{j=1}^{c}1/V_j} \qquad (7.1.3)$$

Para determinar la clase a la que pertenece la muestra $\vec{x}$ se definen tantos volúmenes como clases existentes. El volumen de la clase C_i es determinado por sus k representantes, los más próximos de la muestra $\vec{x}$. La clase ganadora C_i^* es la que posee el volumen más pequeño, es decir, la distancia más pequeña entre la muestra $\vec{x}$ y los prototipos de la clase C_i.

$$\text{Volumen} = \frac{4\pi r^n}{3}=\frac{\pi d^n}{6} \qquad (7.1.4)$$

En el caso de dos clases C_1 y C_2 si $V_2/V_1>1$ o en distancias $D_2>D_1$, entonces la clase ganadora será C_1 y en el caso contrario, la muestra $\vec{x}$ será asignada a la clase C_2. Como se puede observar en la figura (7.1.4.), $V1<V2 \Rightarrow$ la clase ganadora es C_1.

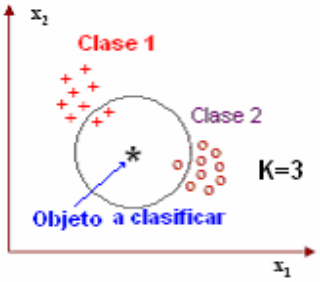


Figura 7.1.3: Modo reagrupamiento general con K=3.

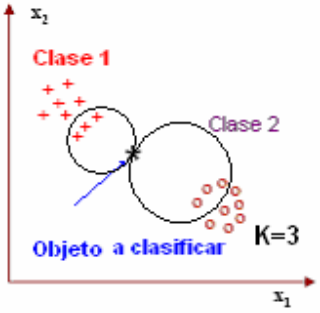


Figura 7.1.4: Modo reagrupamiento por clase (K=3).

Cálculos de distancias

Para calcular la distancia entre la muestra $\vec{x}$ y los puntos de la base de datos M existen diferentes formas de medirla:

Distancia Euclidiana: $$D(\vec{x},\vec{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \qquad (7.1.5)$$

Distancia Manhattan: $$D(\vec{x},\vec{y}) = \sum_{i=1}^{n}(x_i - y_i)^2 \qquad (7.1.6)$$

Distancia del máximo: $$D(\vec{x},\vec{y}) = \max_i |x_i - y_i| \qquad (7.1.7)$$

La distancia que se utiliza normalmente es la euclidiana, pero la distancia *Manhattan* y la del máximo son más rápidas de calcular. El tipo de distancia a utilizar depende de la aplicación, es decir de factores como: el tiempo de ejecución, el costo, el desempeño, la precisión, la portabilidad, etc.

Tipos de rechazo

En distancia: Una muestra $\vec{x}$ es rechazada en distancia cuando ésta se sitúa lejos de la nube principal de puntos. En términos de probabilidad, la muestra es rechazada cuando $p(\vec{x}/C_i)$ es inferior a un cierto umbral α.

En ambigüedad: Una muestra $\vec{x}$ es rechazada cuando su probabilidad de pertenecer a la clase C_i es próxima de la otra clase C_j (con $i \neq j$). La diferencia de probabilidad entre la clase ganadora y la segunda sea inferior a un cierto umbral (*ver figura 7.1.5*).

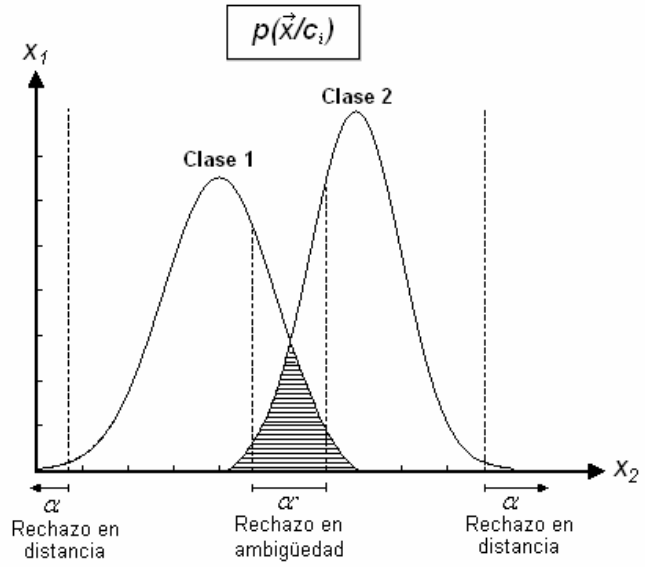


Figura 7.1.5: Tipos de rechazo para la dos clases.

El parámetro K

Un punto crítico para la configuración del clasificador *"k-próximos vecinos*", es la elección de *"K"*, este parámetro es elegido generalmente de una forma heurística. Desde el punto de vista de un estimador de densidad local y para obtener una estimación fiable de $p(C_i, \vec{x})$, $\hat{p}(i, \vec{x})$, la esperanza matemática de $\hat{p}(i, \vec{x})$, debe ser asimptoticamente no sesgada. Por otra parte, la varianza de $\hat{p}(i, \vec{x})$ debe ser minimal. Estas dos condiciones anteriores son alcanzadas si $K \to \infty$ y *N* es el número de muestras es lo mas grande posible. En la práctica se sugiere que $K = \sqrt{N}$. Otra alternativa es buscar el valor de *"K"* en función de la aplicación especifica, es decir, hacer variar *"K"* desde 1 hasta un cierto valor e interpretar los resultados para la *"K"* optima.

Ejemplo de clasificación utilizando con"K" vecinos más próximos.

Para diferenciar tres clases de frutas (C_1=fresas, C_2=naranjas y C_3=melones), con dos rasgos característicos (x1=área y x2=color rojo) y para un punto $\vec{x}$ a clasificar (ec.7.1.8), se toma la base de conocimiento "M" que contiene 27 frutas ordenadas por clases (ec.7.1.9), donde las primeras 9 columnas pertenecen a la clase 1, de la 10 a la 18 a la clase 2 y de la 19 a la 27 a la clase 3. Se procede a calcular el vector de distancias del punto a clasificar a cada una de las columnas de la base de conocimiento (ec.7.1.10). Finalmente las primeras K distancias más próximas y la clase ganadora será la que tenga mas elementos. Para el ejemplo de la figura (7.1.6), si tomamos K=5 se obtiene que la clase a la que pertenece el punto a clasificar es la clase 2 (*naranjas*), ya que tiene 4 vecinos más próximos y la clase 1 (*fresas*) solo tiene un vecino.

$$\vec{X} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{7.1.8}$$

$$\mathrm{M} = \begin{bmatrix} m_{11} & m_{12} & \dots & m_{i1} \\ m_{12} & m_{22} & \dots & m_{i2} \end{bmatrix} \tag{7.1.9}$$

$$D = \begin{bmatrix} d_1 & d_2 & . & . & . & d_i \end{bmatrix} \tag{7.1.10}$$

Figura 7.1.6: Clasificación con K=5 vecinos más próximos, (reagrupamiento general).

7.2. Cálculo de la distancia del objeto

Otro de los objetivos del proyecto, además del reconocimiento del objeto, es el cálculo de la distancia a la que se encuentra el objeto con respecto de la cámara (*webcam*). El algoritmo es muy simple y muy eficiente, lo que cuesta un poco de trabajo es la calibración de la cámara y el láser.

Cómo se muestra en la figura (7.2.1.), el láser es proyectado hacia un objeto que se encuentra en el campo de visión de la cámara a un ángulo "α". La distancia "D" se calcula mediante funciones trigonométricas muy simples y aplicando algunas técnicas del procesamiento digital de imágenes como la resta de imágenes, la erosión y dilatación (*ver capítulo 2*).

Para calibrar la WebCam es necesario ajustar el centro de la imagen proyectada por la cámara con respecto al nivel vertical del centro de la misma, es decir, para una imagen con resolución de 640 x 480 píxeles, el punto (320,240) debe de estar exactamente debajo del centro de la WebCam. Otra cuestión importante es colocar el láser con respecto al centro de visión de la WebCam a 12.7 cm. y ajustar el láser a 70 grados de inclinación para que proyecte el rayo de luz exactamente en el centro de la imagen cuando el objeto se encuentre a una distancia de 35 cm. De esta forma, nuestro sistema será capaz de calcular otras distancias inferiores a 35 cm., como por ejemplo 25 cm. y 30 cm.

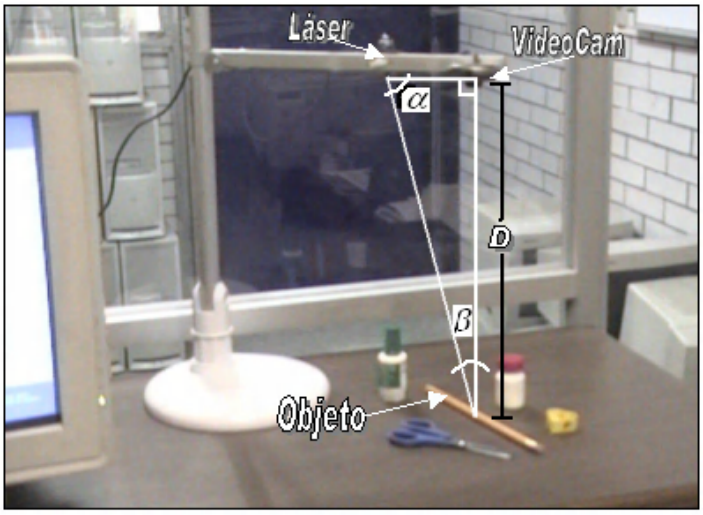


Figura 7.2.1: Calculo de la distancia.

Fundamentos Matemáticos

Función Tangente.- La tangente del ángulo "β" en un triangulo rectángulo es igual al cateto opuesto entre el cateto adyacente (*ver figura 7.2.2*).

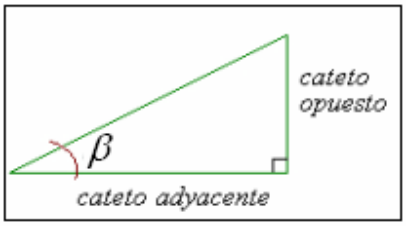
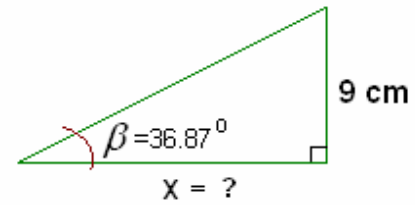


Figura 7.2.2: Triángulo rectángulo.

$$\tan \alpha = \frac{cateto\ opuesto}{cateto\ adyacente} \qquad (7.2.1)$$

Conociendo el ángulo y una de las distancias (cateto opuesto o el cateto adyacente) se puede calcular la otra distancia del triángulo, ejemplo: del siguiente triángulo rectángulo, determina el cateto adyacente.



Respuesta: conocemos el ángulo "β" y el cateto opuesto y aplicando la ecuación (7.2.1) se determina el cateto adyacente "*X*":

$$\tan\beta = 0.75 = \frac{9}{x}$$

$$x = \frac{9}{0.75} = 12\,cm$$

Capítulo 8

GRABACIÓN DE ARCHIVOS DE AUDIO

La etapa final de nuestro proyecto es la reproducción en forma audible del nombre del objeto y la distancia a la que se encuentra con respecto de la cámara (WebCam).

La grabación de archivos de audio se realiza desde un micrófono en formato "wav" por medio de un programa muy fácil de utilizar llamado "Cool Edit" [19]; además con este programa también se pueden, crear efectos de audio, mezclas, arreglar la calidad del audio, unir segmentos de archivos, aumentar o disminuir volumen, entre otras cosas.

La reproducción de los archivos de audio se realiza directamente con *MATLAB* utilizando la función "wavplay".

Antes de presentar la forma de trabajar con Cool Edit, es necesario presentar las características del formato "wav", algunos otros reproductores de audio y las diferentes páginas en Internet de donde se pueden bajar gratuitamente los programas.

8.1 Formatos de sonido digital “WAV”

Este tipo de formato es de los más utilizados en las PCs y en internet. Se pueden hacer grabaciones tipo CD a 44,100 Hz ya sea monoural o estéreo (*Ver tabla 8.1.1*). Este tipo de formato fue desarrollado por Microsoft para utilizarse en ambiente Windows pero existen reproductores de archivos “wav” para casi todas las plataformas existentes.

Un archivo de sonido sin comprimir puede ocupar mucho espacio en su disco duro. Por ejemplo, un minuto de sonido estéreo grabado con un muestreo de 44 KHz ocupará aproximadamente 10.5 MegaBytes. Se pueden grabar archivos de audio en formato en WAV a 16 bits (*65,536 niveles de volumen*) y después grabarlos en un CD por medio de una grabadora de CDs y reproducirlos en un estéreo o reproductor de CDs para música.

8,000 Hz	*Teléfono*
11,000 Hz	*Radio AM*
22,050 Hz	*Radio FM*
44,100 Hz	*CD de sonido*
48,000 Hz	*DAT (Digital Audio Tape)*

Tabla 8.1.1: Diferentes Calidades de Grabación

8.2 Software de edición de sonido

Existen editores y reproductores de sonido disponibles en el mercado como el SoundEdit de Macromedia [26], CoolEdit [27], CakeWalk Pro Audio[28], Sound Forge[29], Qsound[30], Winamp[31], Audacity[32] entre otros.

Además de poder grabar y reproducir archivos de audio en distintos formatos de audio, estos programas tienen opciones muy importantes para diseño como efectos de sonido, mezcladora de sonidos y muchas otras herramientas de gran importancia.

Por medio de la línea de entrada de la tarjeta de sonido se pueden realizar mezclas desde diferentes medios (*televisión, la radio, cassettes, micrófono etc.*); o directamente desde archivos MIDI o WAV.

Algunos de estos programas se encuentran disponibles en la red y se pueden bajar de sitios como:

http://www.topshareware.com/SoundEdit-Pro-transfer-1450.htm
http://ftp.syntrillium.com/pub/cool_edit/ce2kmain.exe
http://www.qsound.com/
http://www.winamp.com/
http://audacity.sourceforge.net/

Para la edición, reproducción y grabación de archivos con formato WAV (*Archivos de ondas*) únicamente nos enfocaremos en el CoolEdit Pro ya que para nuestro proyecto, cubre con los elementos necesarios en la grabación de archivos de audio.

8.3 CoolEdit Pro

Es una herramienta de edición de sonidos, con ella se puede grabar, reproducir y editar archivos de ondas de 8 bits (*calidad de cinta*) y de 16 bits (*calidad de CD*) y hasta 32 bits. Con CoolEdit Pro, puede grabar sonidos de diversas fuentes para luego manipularlos a su gusto, cortar y pegar segmentos de archivos de ondas (*WAV*) para combinar sonidos y añadir efectos especiales como reverberación, eco y desvanecimiento, entre otros. También reproducir un fragmento musical de un CD, un mensaje hablado por el micrófono y juntarlos en un archivo para utilizarlo en una presentación audiovisual.

Grabación desde un CD o desde un micrófono.

Para grabar canciones de un CD a un archivo “WAV”, lo primero que necesita es crear un nuevo archivo seleccionando <New> del menú <File>. Aparece una ventana en donde debe seleccionar la frecuencia, el número de canales y los bits a los que se grabará el nuevo archivo (como se muestra en la figura 8.3.1). El segundo paso es activar las herramientas del “CD” seleccionando la opción <Show CD Player> dentro del menú <View> y presionar el botón <Record> (Grabar), finalmente debe seleccionar la pista que desea escuchar y presionar el botón de <Play> (ver figura 8.3.2). Todo lo que escucha en las bocinas se esta almacenando en el buffer de la memoria de la computadora hasta que presione el botón de <Stop>. Para almacenar nuestro archivo en el disco duro será necesario seleccionar <Save> dentro del menú <File>, ver figura (8.3.3).

Los menús gráficos son parecidos a los símbolos de las grabadoras de video o de los estéreos caseros; además, para evitar confusiones, si desplaza el cursor por encima de un botón del menú aparecerá en un recuadro el nombre de la acción que realiza.

Si deseamos grabar desde un micrófono lo único que tenemos que hacer es presionar el botón <Record> de la grabadora de Cool Edit y presionar el botón de <Stop> para terminar la grabación. Una cuestión importante es tener activado el micrófono desde el control de volumen de Windows y verificar que funcione adecuadamente el controlador de audio.

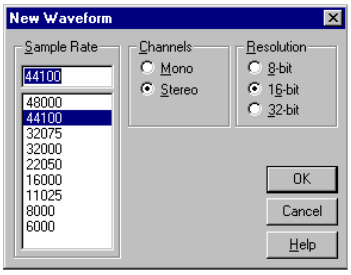


Figura 8.3.1. Calidad de grabación.

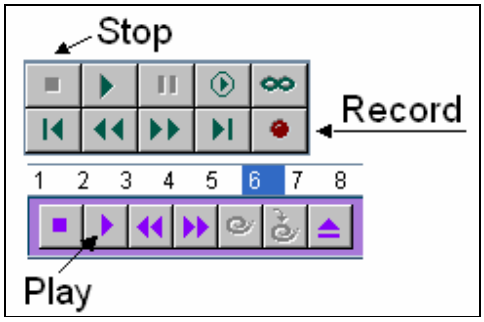


Figura 8.3.2. Botones de Cool Edit y del CD.

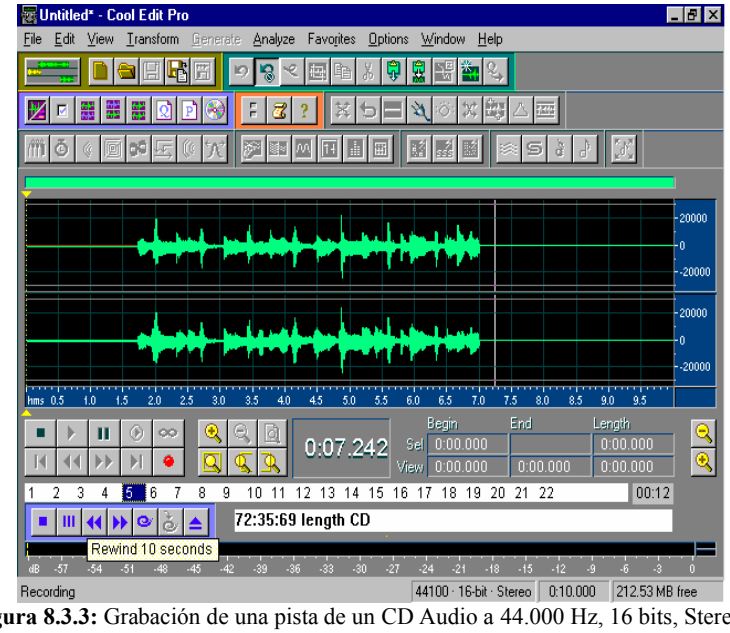


Figura 8.3.3: Grabación de una pista de un CD Audio a 44.000 Hz, 16 bits, Stereo.

Cambios de volumen

Si deseamos una reducción continua de volumen en una parte del archivo utilizaremos la herramienta <Fade Out> que se encuentra dentro del menú <Amplify>, seleccionando la zona en que deseamos reducir el volumen y aplicar esta herramienta.

En las figuras (8.3.4.) y (8.3.5.) se aprecia claramente la selección de pistas y como queda el nuevo archivo final. Es importante que la zona seleccionada sea de al menos de 8 segundos, porque de lo contrario se obtiene un final muy precipitado.

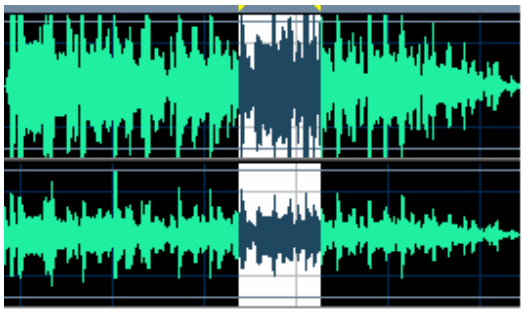


Figura 8.3.4: Selección para utilizar como fin de pista.

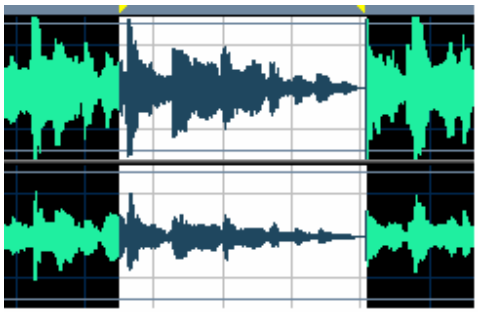


Figura 8.3.5: Después de cortar el final, con Fade Out se obtiene el nuevo final.

Eliminación de ruidos “Hiss Reduction”

En algunas grabaciones es necesario eliminar picos de señal que distorsionan el sonido. Hiss Reduction es un filtro ideal que elimina este molesto ruido continuo.

Para eliminar el ruido de fondo es necesario seleccionar una sección del archivo que sólo contenga ruido de fondo, como se muestra en la figura 8.3.6. Estas zonas normalmente se encuentran al principio o al final de la canción. Entonces se abre el menú de <Noise Reduction> (ver figura 8.3.7) y se emplea la función <Get Profile From Selection> con un valor de <Snapshots> mínimo de 84. Los tramos iniciales o finales que siempre contienen ruido de fondo pueden ser limpiados con la opción de cortar y para recuperar esos segundos que debe haber entre pista y pista, se deben insertar dos segundos de silencio digital mediante la función <generate silence>, aunque por lo general los programas de grabación de CD Audio ya insertan silencio digital entre pistas.

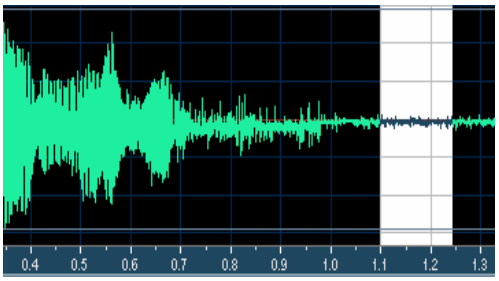


Figura 8.3.6: Ruido de fondo en la señal de onda.

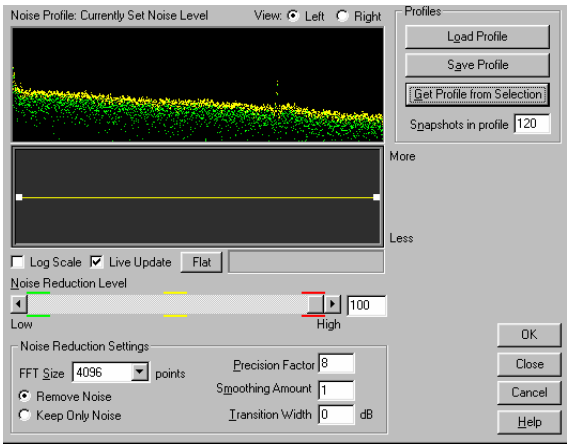


Figura 8.3.7: Filtro *“Noise Reduction”*.

Capítulo 9

IMPLEMENTACIÓN DEL PROYECTO

En este capítulo se presentan los resultados obtenidos en cada una de las etapas del sistema de reconocimiento automático de objetos *RAO*, desde la captación de la imagen hasta la etapa de reconocimiento e interpretación, así como la grabación de archivos de audio.

Se realizaron pruebas sobre el tipo de resolución a emplear, se analizaron algunos puntos muy importantes en la captación de imágenes, la manera de aplicar filtros para eliminación de ruido en la imagen, se implementó un método para la segmentación de imágenes en un fondo negro, se analizaron los diferentes descriptores de objetos y se aplicó el clasificador "*K próximos vecinos*" de forma exitosa.

Se muestran también el código fuente de algunas rutinas realizadas en Matlab versión 7.0 mediante algoritmos y fórmulas que se presentaron en capítulos anteriores.

9.1. Resultados en la adquisición de la imagen

En la etapa de Adquisición de la imagen se utilizó una VideoCam Genius NB y algunas herramientas de adquisición de imágenes "*Image Acquisition Toolbox*" y de procesamiento de imágenes ("*Image Processing Toolbox"*) del lenguaje MATLAB versión 7.0 (ver apéndice A).

Las imágenes se capturaron en formato BMP por ser un formato que no tiene compresión y por lo tanto no distorsiona los niveles de color. En la figura 9.1.1. se muestra la VideoCam utilizada y en la figura (9.1.2) los resultados la imagen captada.

Figura 9.1.1: VideoCam Genius NB.

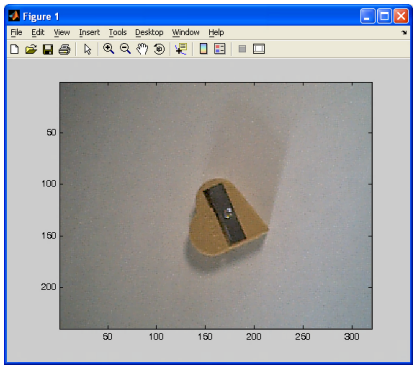


Figura 9.1.2: Captación de la imagen a través de la VideoCam.

La resolución que se utilizó para este sistema de reconocimiento es de 640x480 píxeles, ya que si se utilizaba una resolución más baja se pierde detalle del objeto a reconocer (*ver figura 9.1.3*).

Figura 9.1.3: Diferentes tipos de resolución.

Uno de los problemas frecuentes en la etapa de captación de la imagen es la inadecuada iluminación del objeto a reconocer (*ver figura 9.1.4.*), ya que es posible que las sombras se tomen como parte del objeto o los cambios de tono del color, hacen que el sistema de reconocimiento no reconozca el color real del objeto.

Para evitar las sombras producidas por la iluminación, nuestro sistema de reconocimiento se implementó dentro de un salón de clases con adecuada iluminación.

Figura 9.1.4: Objeto a reconocer con una mala iluminación.

Otros puntos que se deben tomar en cuenta en la etapa de captación de la imagen son: el tamaño del objeto, el ángulo de captura, el enfoque de la VideoCam, entre otros. Si nuestro objeto es muy grande, la VideoCam no alcanza a captar todo el objeto; y si la cámara no está bien enfocada, el objeto aparece muy borroso y no se puede determinar la identidad del objeto (*ver figura 9.1.5*). Para evitar este tipo de detalles, se coloco la VideoCam a 20 cm. del objeto, se realizo un buen enfoque del objeto y se utilizaron objetos pequeños que no rebasaran el ángulo de visión de la VideoCam.

A) B) C)

Figura 9.1.5: Consideraciones importantes en la captación de imágenes:
A) Tamaño del objeto, B) ángulo de captación y C) desenfoque.

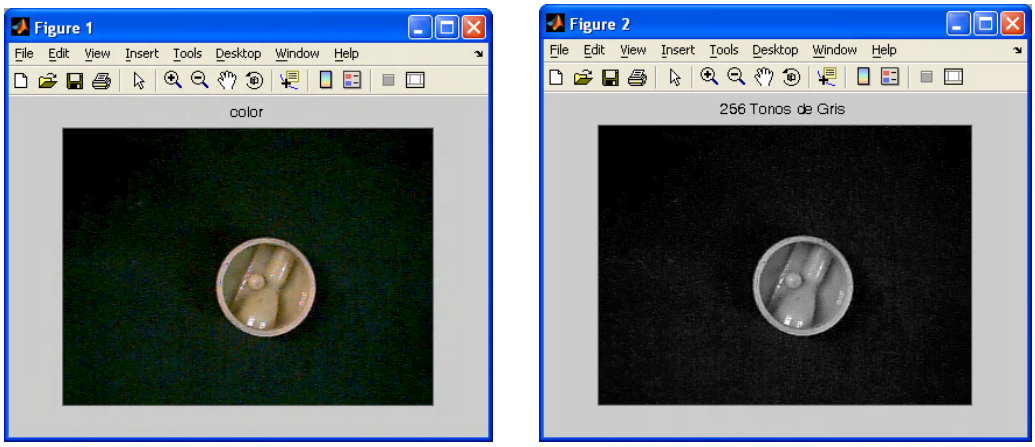
El código para la captura de imágenes a través de la VideoCam se muestra en el programa #1, donde se utiliza el dispositivo número 2 "*Genius VideoCAM NB*" con un formato de captura en color "RGB24_640x480".

```
clear all
video = videoinput('winvideo', 2,'RGB24_640x480');
preview(video)
x=input('ENTER PARA CAPTURAR IMAGEN');
imagen = getsnapshot(video);
imagesc(imagen)
imwrite(imagen,'sacapuntas1.bmp','bmp');
delete(video)

```

Programa #1 Captación de la imagen

9.2. Resultados en el preprocesamiento de la imagen

Antes de realizar mejoras a la imagen es necesario hacer una conversión a niveles de gris y así poder eliminar o suavizar un poco el ruido en la imagen obtenida por la VideoCam. Para realizar la conversión a niveles de gris se utilizó el promedio de las tres matrices de cada píxel para cada color RGB (*ver figura 9.2.1*).



A) B)

Figura 9.2.1. Conversión de una imagen en color a blanco y negro: A) imagen en color, B)imagen en niveles de gris.

La codificación del programa de conversión a niveles de gris se muestra en el programa #2. Donde se separan cada uno de los componentes en RGB de la imagen en color y se realiza el promedio de los tres colores por cada píxel.

```
1  clear; close all
2  im_color = imread('sacapuntas1.bmp');
3  rojo=double(im_color(:,:,1));
4  verde=double(im_color(:,:,2));
5  azul=double(im_color(:,:,3));
6  im_gris=round((rojo(:,:)+verde(:,:)+azul(:,:))/3);
7  im_gris=uint8(im_gris);
8  figure, imshow(im_color); title('color');
9  figure, imshow(im_gris); title('256 Niveles de Gris');
10
```

Programa #2 Conversión a niveles de gris

Se utilizaron diferentes tipos de filtros descritos en el capítulo 4 como son: *el filtro promedio, mediana, mínimo, máximo y punto medio*; pero el filtro con el que se obtuvieron mejores resultados para nuestro proyecto fue el filtro promedio ya que además de eliminar un poco el ruido, realiza un suavizado de la imagen, como se muestra en la figura (9.2.2.).

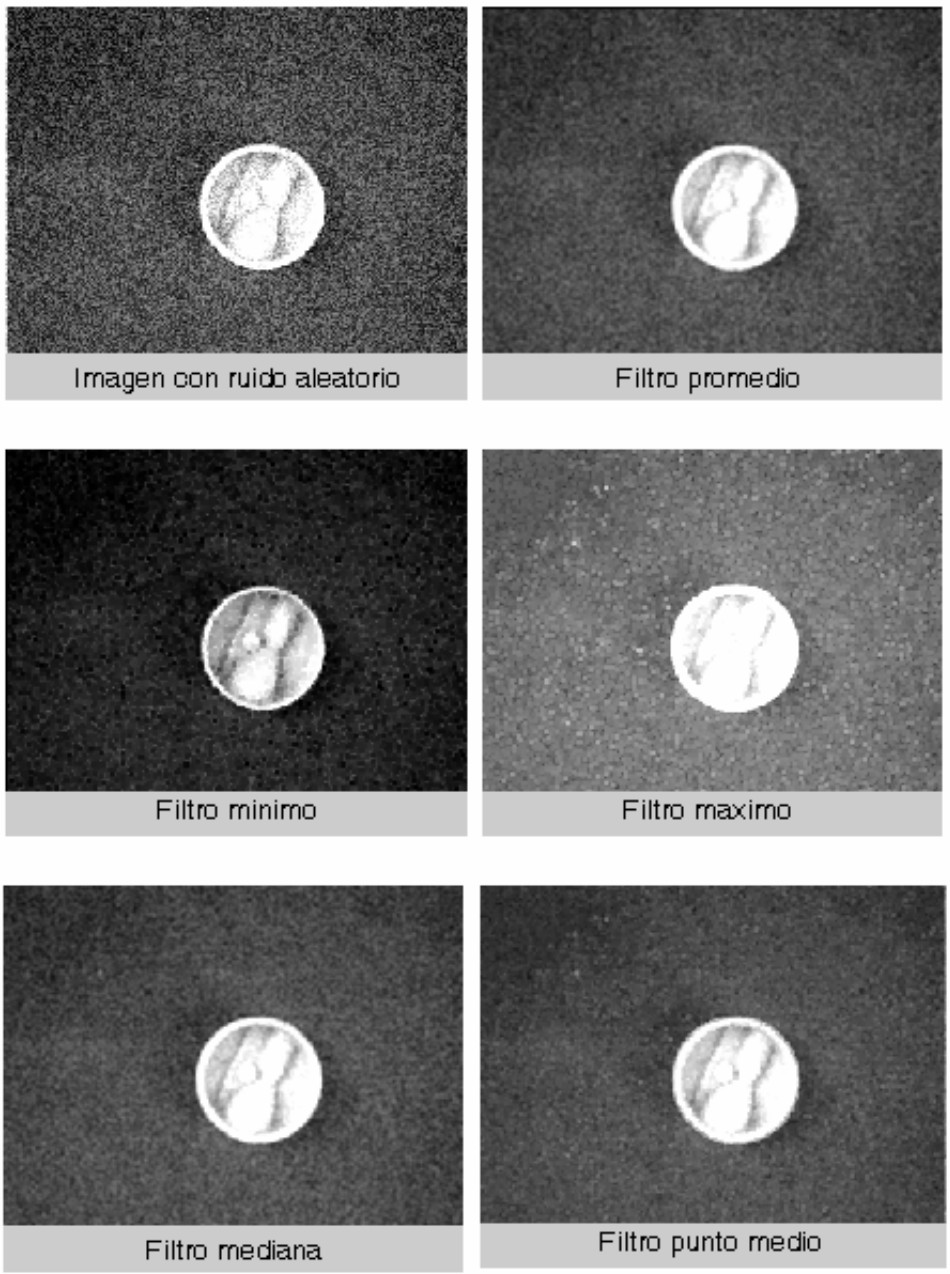


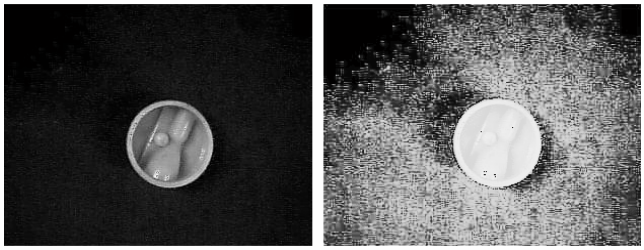
Figura 9.2.2: Uso de filtros para una imagen con ruido aleatorio.

El programa #3 ejemplifica el uso de diferentes filtros a una imagen a la cual se le aplicó ruido aleatorio. Este programa se implementó con máscaras de 3x3 (*ver capítulo 4*), comenzando con la segunda fila y segunda columna de la imagen para terminar con la penúltima fila y penúltima columna de la misma imagen; pero la imagen resultante para cada filtro tiene dos columnas y dos filas menos que la original.

```
clear all; close all;
im_color=imread('sacapuntas1.bmp');
rojo=double(im_color(:,:,1));
verde=double(im_color(:,:,2));
azul=double(im_color(:,:,3));
im_gris=round(rojo(:,:)*0.33)+round(verde(:,:)*0.59)+round(azul(:,:)*0.11);
im=double(im_gris);

ruido_aleatorio=floor(rand(fil,col)*100);
im_ruido=im+ruido_aleatorio;
im=im_ruido;
[fil col]=size(im);
for x=2:1:fil-1
    for y=2:1:col-1
        a=im(x-1,y-1);b=im(x,y-1);c=im(x+1,y-1);d=im(x-1,y);e=im(x,y);f=im(x+1,y);
           g=im(x-1,y+1);h=im(x,y+1);i=im(x+1,y+1);
        im_fil_prom(x,y)=round((a+b+c+d+e+f+g+h+i)/9);
        vector(1:9)=sort([a b c d e f g h i]);
        im_fil_mediana(x,y)=round(sum(vector)/9);
        im_fil_minimo(x,y)=vector(1);
        im_fil_maximo(x,y)=vector(9);
        im_fil_punto_m(x,y)=round((vector(1)+vector(9))/2);
     end
end

 figure
 hold on
subplot(2,3,1); imshow(uint8(im_ruido));title('Imagen con  ruido aleatorio');
subplot(2,3,2); imshow(uint8(im_fil_prom));title('Filtro promedio');
subplot(2,3,3); imshow(uint8(im_fil_mediana));title('Filtro mediana');
subplot(2,3,4); imshow(uint8(im_fil_minimo));title('Filtro minimo');
subplot(2,3,5); imshow(uint8(im_fil_maximo));title('Filtro maximo');
subplot(2,3,6); imshow(uint8(im_fil_punto_m));title('Filtro punto medio');

```

Programa #3: Filtros para imágenes en niveles de gris.

Otro programa que se realizó para el mejoramiento de la imagen fue el de ampliación de histograma, el problema que se encontró aquí, es que también resalta el ruido y sombras (*ver figura 9.2.3*); por lo tanto, no se utilizó esta técnica en la implementación de nuestro sistema de reconocimiento. El código para realizar esta técnica se presenta en el programa # 4.



A) B)

Figura 9.2.3. Ampliación de histograma A) imagen original, B) imagen resultado de la ampliación de histograma.

```
im=double(im_gris);
[fil col]=size(im);
vectorG=zeros(1,256);
for i=1:1:256; vectorG(i)=i-1; end;
vectorF=double(zeros(1,256));
vectorFa=double(zeros(1,256));
vectorFg=double(zeros(1,256));
im=double(im_gris);
for i=1:1:256
   for x=1:1:fil
     for y=1:1:col
         in=double(im(x,y));
         if i==in vectorF(i)=vectorF(i)+1; end;
     end
   end
end
vectorFa(1)=vectorF(1);
 for i=2:1:256 vectorFa(i)=vectorFa(i-1)+vectorF(i); end;
 for i=1:1:256
     vectorFg(i)=round((256*vectorFa(i))/(fil*col))-1;
     if vectorFg(i)<0 vectorFg(i)=0; end
 end
im_res=zeros(fil,col); im=double(im_gris);
for i=1:1:256
 for x=1:1:fil
  for y=1:1:col
      in=double(im(x,y));
      if in==vectorG(i)&& im_res(x,y)<in im_res(x,y)=vectorFg(i); end
  end
 end
end
figure; imshow(uint8(im_gris));title('Imagen gris');
figure; imhist(uint8(im_gris));title('histograma');
figure; imshow(uint8(im_res));title('resultado');
```

Programa #4: Ampliación de histograma.

9.3. Resultados en la segmentación

En la etapa de segmentación, se utilizó la binarización con diferentes umbrales para aislar la imagen del fondo y eliminar el ruido. Las pruebas que se realizaron se muestran en la figura 9.3.1. En algunos casos se forman manchas de color negro dentro de la imagen y en otras se agregan pequeños puntos blancos fuera de la imagen; pero para la implementación de nuestro sistema se utilizo el método de Otsu que realiza la binarización en forma automática.

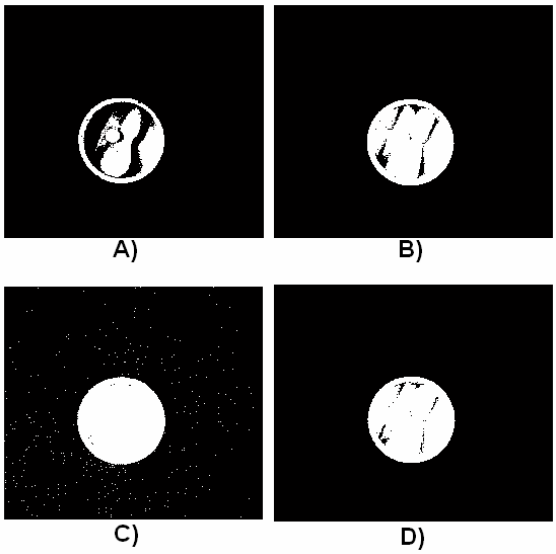


Figura 9.3.1. Binarización de la imagen; A) umbral = 192, B) umbral = 128, C) umbral = 64, D) umbral = 73 (umbral de Otsu para este caso).

El problema de la binarización es encontrar un umbral óptimo que divida la imagen a blanco y negro sin que se agregue ruido a la imagen a segmentar. El programa #5 muestra como binarizar una imagen en niveles de gris a blanco y negro con diferentes umbrales y en el programa #6 se muestra la función “graythresh” que utiliza el método de Otsu para calcular el umbral óptimo de forma automática para una imagen en color.

```
[ancho largo]=size(im_gris)
for x=1:1:ancho
    for y=1:1:largo
        intensidad=im_gris(x,y);
        im_negativo(x,y)=uint8(255-double(intensidad));
        if intensidad>192 im_byn1(x,y)=255; else im_byn1(x,y)=0; end
        if intensidad>128 im_byn2(x,y)=255; else im_byn2(x,y)=0; end
        if intensidad>64 im_byn3(x,y)=255; else im_byn3(x,y)=0; end
    end
end
figure, imshow(im_color); title('color');
figure, imshow(im_gris); title('256 Niveles de Gris');
figure, imshow(im_byn1);title('blanco y negro 1');
figure, imshow(im_byn2);title('blanco y negro 2');
figure, imshow(im_byn3);title('blanco y negro 3');

```

Programa #5: Segmentación con diferentes umbrales

```
clear all; close all
im_color=imread('sacapuntas1.bmp');
nivel = graythresh(im_color);
im_byn = im2bw(im_color,nivel);
figure, imshow(im_byn);title('imagen en blanco y negro Otsu');

```

Programa #6: Segmentación con el método de Otsu

En la figura (9.3.2.) se muestran dos técnicas para eliminación de ruido mediante operaciones morfológicas sobre imágenes binarias. La cerradura seguida de una apertura une los puntos blancos que están casi unidos, eliminando pequeños puntos negros; La apertura seguida de la cerradura disuelve casi por completo los puntos blancos y se obtienen mejores resultados.

A) B) C)

Figura 9.3.2 Operaciones morfológicas: A) imagen binarizada con ruido, B) aplicación de cerradura seguida de una apertura, C) aplicación de una apertura seguida de una cerradura.

Después de aplicar la cerradura seguida de una apertura, es necesario eliminar los huecos que se encuentran en el interior del objeto utilizando la instrucción de MatLab *"imfill"*. También será necesario eliminar pequeños objetos que por algún motivo se filtraron; para ello se utilizó la instrucción *"bwareaopen"*.
A continuación se presenta su implementación:

```
1   im = im_open_close;
2   im = imfill(im,'holes');
3   im = bwareaopen(im,500);
```

donde, en la primera línea se almacena la imagen resultante de aplicar la apertura y cerradura en la imagen "*im*", la segunda línea elimina los huecos de la imagen "*im*" y la tercera línea elimina elementos menores a 500 píxeles de la imagen "*im*". En la figura 9.3.3 se presenta el resultado obtenido al utilizar las funciones anteriores.

A) B)

Figura 9.3.3 Eliminación de huecos y pequeños objetos: A) imagen con huecos y pequeños puntos, B) Resultado de aplicar las funciones "imfill" y "bwareaopen".

El siguiente paso fue aislar por completo al objeto para determinar las coordenadas de cada punto perteneciente al objeto y así calcular su área, perímetro, color, textura, etc. y realizar el reconocimiento. La forma en que se realizó el proceso fue multiplicando cada valor de la matriz correspondiente a cada color de la imagen por cada valor de la imagen segmentada, como se muestra a continuación:

```
1   im=uint8(im);
2   im_color2(:,:,1)=im_color(:,:,1).*im;
3   im_color2(:,:,2)=im_color(:,:,2).*im;
4   im_color2(:,:,3)=im_color(:,:,3).*im;
```

donde, "*im_color*" es la imagen original en color, "*im_color2*" es la imagen destino y "*im*" es la imagen binaria. Antes de realizar el proceso es necesario convertir la imagen binaria "*im*" a formato entero de 8 bits. El resultado de este proceso se muestra en la figura 9.3.4.

A) B) C)

Figura 9.3.4 Multiplicación de matrices punto a punto: A) imagen en color, imagen en blanco y negro, C) imagen resultado de la multiplicación.

También se hicieron pruebas con diferentes tipos de algoritmos para la detección de contornos como los que se muestran en la figura 9.3.5. Uno de los algoritmos mas simples y que dio muy buenos resultados fue el de *Roberts* en su versión simplificada (*ver capítulo 4*). Cabe mencionar que en el presente proyecto no se utilizó el contorno del objeto ya que para la detección del perímetro se utilizó otra técnica. El código para detección de contornos se muestra en el programa # 7.

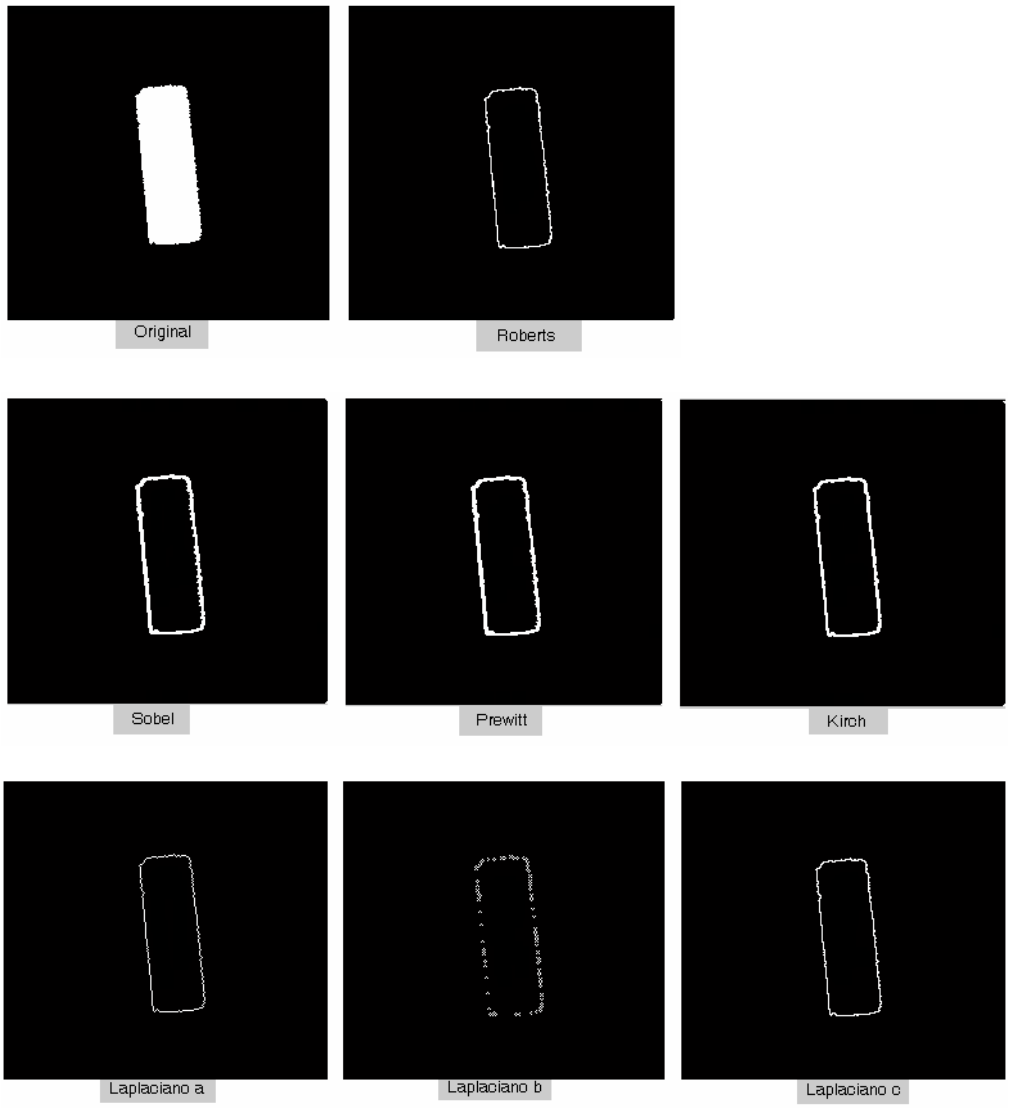


Figura 9.3.5 Aplicación de diferentes algoritmos para la detección de contornos

```
im=double(im);
[ancho largo]=size(im)
for x=2:1:ancho-1
    for y=2:1:largo-1
        a=im(x-1,y-1);b=im(x,y-1);c=im(x+1,y-1);d=im(x-1,y);e=im(x,y);
        f=im(x+1,y); g=im(x-1,y+1);h=im(x,y+1);i=im(x+1,y+1);
        a=double(a);b=double(b);c=double(c);d=double(d);e=double(e);
        f=double(f);g=double(g);h=double(h);i=double(i);
        Roberts(x,y)=abs(e-a)+abs(b-d);
        SobelM1=a*(-1)+d*(-2)+g*(-1)+c+f*2+i;
        SobelM2=a*(-1)+b*(-2)+c*(-1)+g+h*2+i;
        valores(1:2)=[SobelM1 SobelM2];
        Sobel(x,y)=uint8(max(valores));
        PrewittM1=a*(-1)+b*(-1)+c*(-1)+g+h+i;
        PrewittM2=c*(-1)+f*(-1)+i*(-1)+a+d+g;
        valores(1:2)=[PrewittM1 PrewittM2];
        Prewitt(x,y)=uint8(max(valores));
        KirchM1=a*(+5)+b*(+5)+c*(+5)+d*(-3)+f*(-3)+g*(-3)+h*(-3)+i*(-3);
        KirchM2=a*(-3)+b*(+5)+c*(+5)+d*(-3)+f*(+5)+g*(-3)+h*(-3)+i*(-3);
        KirchM3=a*(-3)+b*(-3)+c*(+5)+d*(-3)+f*(+5)+g*(-3)+h*(-3)+i*(+5);
        KirchM4=a*(-3)+b*(-3)+c*(-3)+d*(-3)+f*(+5)+g*(-3)+h*(+5)+i*(+5);
        KirchM5=a*(-3)+b*(-3)+c*(-3)+d*(-3)+f*(-3)+g*(+5)+h*(+5)+i*(+5);
        KirchM6=a*(-3)+b*(-3)+c*(-3)+d*(+5)+f*(-3)+g*(+5)+h*(+5)+i*(-3);
        KirchM7=a*(+5)+b*(-3)+c*(-3)+d*(+5)+f*(-3)+g*(+5)+h*(-3)+i*(-3);
        KirchM8=a*(+5)+b*(+5)+c*(-3)+d*(+5)+f*(-3)+g*(-3)+h*(-3)+i*(-3);
        valores(1:8)=[KirchM1 KirchM2 KirchM3 KirchM4 KirchM5 KirchM6 KirchM7 KirchM8];
        Kirch(x,y)=uint8(max(valores));
        LaplacianoM1(x,y)=b*(-1)+d*(-1)+e*(4)+f*(-1)+h*(-1);
        LaplacianoM2(x,y)=a+b*(-2)+c+d*(-2)+e*(4)+f*(-2)+g+h*(-2)+i;
        LaplacianoM3(x,y)=a*(-1)+b*(-1)+c*(-1)+d*(-1)+e*(8)+f*(-1)+g*(-1)+h*(-1)+i*(-1);
   end
end
figure, imshow(im) ;  title('Original');
figure, imshow(Roberts);title('Roberts');
figure, imshow(Sobel);title('Sobel');
figure, imshow(Prewitt);title('Prewitt');
figure, imshow(Kirch);title('Kirch');
figure, imshow(LaplacianoM1);title('Laplaciano a');
figure, imshow(LaplacianoM2);title('Laplaciano b');
figure, imshow(LaplacianoM3);title('Laplaciano c');
```

Programa #7: Detección de contornos

9.4. Resultados en la representación y descripción

Los rasgos característicos utilizados en este proyecto son el color, el factor de compacidad, la excentricidad y la energía. Estos rasgos tienen la característica de ser invariantes a rotaciones y a cambios de escala. Para fines de la clasificación se utilizaron valores normalizados entre 0 y 1. En la tabla (9.4.1.) se puede observar que se utilizan 4 objetos de color blanco y 4 objetos de color amarillo para que de esta forma tratar de confundir al primer clasificador (*clasificación por color*). El segundo descriptor es la forma del objeto (*factor de compacidad y excentricidad*), finalmente, si todavía se tienen dudas sobre de qué objeto se trata, se utiliza el descriptor de textura (*energía*).

#	OBJETO	COLOR
1	*Lápiz*	*Blanco*
2	*Pluma*	*Amarillo*
3	*Sacapuntas*	*Amarillo*
4	*Diurex*	*Amarillo*
5	*Borrador*	*Amarillo*

#	OBJETO	COLOR
6	*Calculadora*	*Gris*
7	*Corrector*	*Blanco*
8	*Goma*	*Blanco*
9	*Resistol*	*Blanco*
10	*Marcador*	*Rosa*

Tabla 9.4.1: Color de los objetos a clasificar.

Se capturaron 24 imágenes por objeto a distancias de 25, 30 y 35 cm. con una rotación de 0, 45, 90, 135, 180, 225, 270 y 315 grados. En total de imágenes capturadas fue de 240 ya que se utilizaron 10 objetos. A cada una de estas imágenes se extrajeron sus rasgos característicos y se almacenaron en la base de conocimiento del sistema para su posterior clasificación. En la tabla 9.4.2. se muestran solo 8 imágenes del objeto 1 (*lápiz*) en diferentes rotaciones y a una distancia de 25 cm. Los rasgos característicos tienen poca variabilidad entre cada imagen ya que se trata del mismo objeto.

OBJETO	ANGULO	DISTANCIA	ROJO	VERDE	AZÚL	FCN	EXCENTRICIDAD	ENERGIA
1	0	25	0.965	0.859	0.800	0.514	0.998	0.395
1	45	25	0.733	0.902	0.945	0.790	0.998	0.322
1	90	25	0.816	0.855	0.886	0.722	0.998	0.299
1	135	25	0.812	0.871	0.878	0.822	0.998	0.342
1	180	25	0.808	0.886	0.894	0.538	0.998	0.374
1	225	25	0.816	0.890	0.922	0.799	0.998	0.378
1	270	25	0.816	0.910	0.941	0.709	0.998	0.460
1	315	25	0.792	0.863	0.878	0.847	0.998	0.321

Tabla 9.4.2: Rasgos característicos del objeto #1 (lápiz).

DESCRIPTOR DE COLOR UTILIZADO

Inicialmente en el presente proyecto se utilizó el espacio de color “HSI” (*Hue, Saturation, Intensity*) que describe el color, pureza del color y brillo (*ver capítulo 6*) y se utilizó un clasificador para determinar el color del objeto. Los colores a clasificar se muestran en la figura (9.4.1.), los cuales se refieren a los colores secundarios. Los valores de color calculados en el espacio de color “HSI” se muestran en la tabla (9.4.3.).

Figura 9.4.1: Colores secundarios.

COLOR	Valor HSI
rojo	0.0105
rojo anaranjado	0.0556
anaranjado	0.1020
amarillo anaranjado	0.1294
amarillo	0.1560
amarillo verdoso	0.1882
verde	0.3333
verde azulado	0.4792
azul	0.6183
azul violeta	0.7503
violeta	0.8201
violeta rojizo	0.9171

Tabla 9.4.3: Valores de color HSI.

Como se puede apreciar, el espacio de color “HSI” ofrece un descriptor de color eficiente ya que las diferencias de los valores de color son discriminantes. Lo que puede variar es la iluminación del objeto en el momento de la captura.

En este caso se esperaba tener muy buenos resultados; pero no fue así, los valores del color rojo varían entre [0.9 y 0.5]. Se puede notar que existe un salto de una unidad de 0.9999 a 0.0000, lo cual causó problemas al momento de la clasificación para el cálculo de las distancias.

El espacio de color que finalmente se utilizo fue el RGB(*Red, Green, Blue*) ya que la combinación de estos tres colores nos discriminan perfectamente 4 de los 10 objetos utilizados y como los objetos no tiene un color uniforme, se optó por utilizar el promedio del color del objeto segmentado para el reconocimiento. La captura de las imágenes se realizó dentro de un salón de clases a medio día (*cuando se tiene una mayor iluminación*). La luz que se filtra por las ventanas y la puerta puede ser determinante ya que modifica el color del objeto (*ver figuras 9.4.2 a y 9.4.2 b*). La posición del objeto con respecto de la fuente de iluminación y los reflejos de luz también causan problemas en el proceso de segmentación.

a) (0.96,0.86,0.80) b) (0.82,0.90,0.91)

Figuras 9.4.2 a) Color promedio del lápiz color blanco con una mala iluminación, b) Color promedio del lápiz con una buena iluminación.

Los datos obtenidos del color promedio para los diez objetos y su desviación estándar se presentan en las tablas (9.4.4), (9.4.5), y (9.4.6.), con sus respectivas gráficas en las figuras (9.4.3.), (9.4.4) y (9.4.5.).

COLOR ROJO				
OBJETO	PROMEDIO	+ DESVEST	- DESVEST	DESVEST
Lápiz	0.844	0.890	0.798	0.046
Pluma	0.934	0.991	0.877	0.057
Sacapuntas	0.807	0.826	0.788	0.019
Diurex	0.534	0.586	0.482	0.052
Borrador	0.639	0.663	0.615	0.024
Calculadora	0.686	0.731	0.641	0.045
Corrector	0.781	0.818	0.744	0.037
Goma	0.657	0.689	0.625	0.032
Resistol	0.804	0.868	0.740	0.064
Marcador	0.954	0.965	0.943	0.011

Tabla 9.4.4: Promedio y desviación estándar para el color rojo.

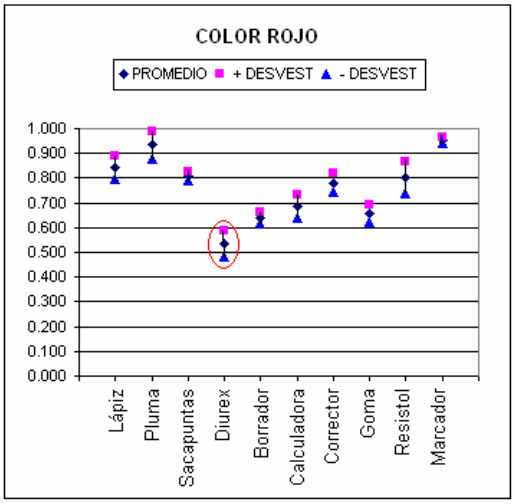


Figura 9.4.3 Gráfica correspondiente al color rojo.

COLOR VERDE				
OBJETO	PROMEDIO	+ DESVEST	- DESVEST	DESVEST
Lápiz	0.879	0.899	0.859	0.020
Pluma	0.578	0.596	0.560	0.018
Sacapuntas	0.589	0.603	0.575	0.014
Diurex	0.554	0.618	0.490	0.064
Borrador	0.487	0.501	0.473	0.014
Calculadora	0.708	0.758	0.658	0.050
Corrector	0.849	0.872	0.826	0.023
Goma	0.636	0.652	0.620	0.016
Resistol	0.908	0.932	0.884	0.024
Marcador	0.335	0.355	0.315	0.020

Tabla 9.4.5: Promedio y desviación estándar para el color verde.

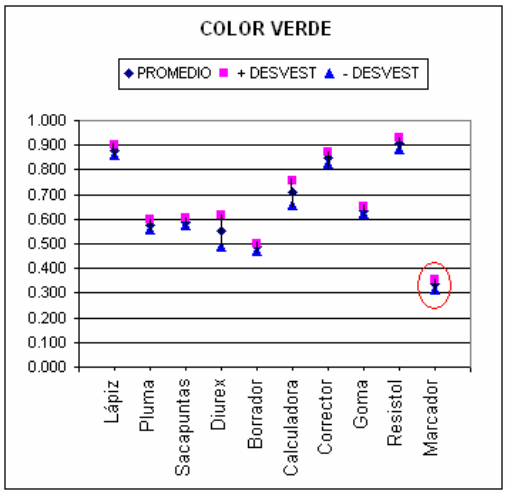


Figura 9.4.4: Gráfica correspondiente al color verde.

COLOR AZUL				
OBJETO	PROMEDIO	+ DESVEST	- DESVEST	DESVEST
Lápiz	0.912	0.945	0.879	0.033
Pluma	0.383	0.439	0.327	0.056
Sacapuntas	0.305	0.315	0.295	0.010
Diurex	0.549	0.621	0.477	0.072
Borrador	0.485	0.500	0.470	0.015
Calculadora	0.716	0.776	0.656	0.060
Corrector	0.867	0.898	0.836	0.031
Goma	0.828	0.858	0.798	0.030
Resistol	0.932	0.962	0.902	0.030
Marcador	0.556	0.587	0.525	0.031

Tabla 9.4.6: Promedio y desviación estándar para el color azul.

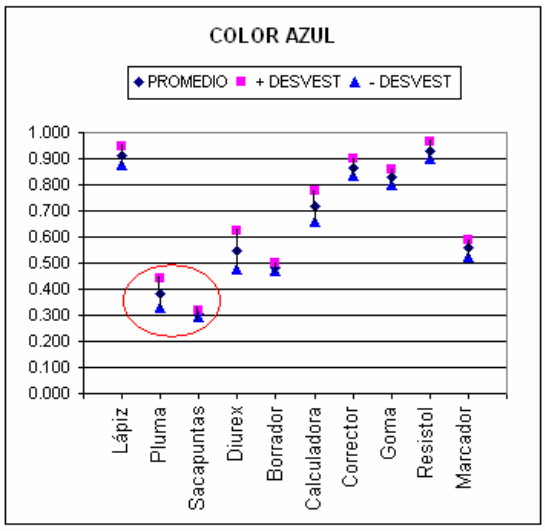


Figura 9.4.5 Gráfica correspondiente al color azul.

De las gráficas obtenidas, se puede observar que el componente de color rojo para el objeto 4 (*diurex*) se encuentra separado del grupo con un valor promedio de 0.53, el cuál se puede discriminar fácilmente. Para el color verde el objeto que se aísla del grupo es el objeto 10 (*marcador*) con un promedio de 0.33 y para el color azul los objetos que se encuentran separados son el lápiz y el sacapuntas con promedios 0.38 y 0.30 respectivamente. La forma más adecuada para obtener mejores resultados es utilizar en conjunto los tres descriptores de color por medio de un clasificador.

DESCRIPTORES DE FORMA UTILIZADOS

Los resultados de utilizar los rasgos característicos de *Factor de Compacidad* y *Excentricidad* a diferentes rotaciones y distancias se muestran en la tabla (9.4.7), como se puede observar para el objeto 3 (*sacapuntas*) que se tomó como muestra, no existen diferencias significativas; por lo tanto, se puede concluir que son buenos descriptores de forma. Estos resultados son posibles gracias a una buena segmentación del objeto y eliminación de ruidos en la imagen.

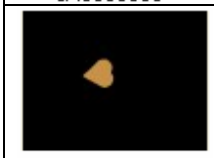
IMAGEN	ROTACIÓN	DISTANCIA	FACTOR DE COMPACIDAD	EXCENTRICIDAD
 IM0300025	0^0	25 cm	0.127	0.260
 IM0300030	0^0	30 cm	0.126	0.248
 IM0300035	0^0	35 cm	0.126	0.245
 IM0304525	45^0	25 cm	0.108	0.269
 IM0304530	45^0	30 cm	0.110	0.246
 IM0304535	45^0	35 cm	0.108	0.254

Tabla 9.4.7: Descriptores de forma a diferentes distancias con rotaciones a 0^o y 45^o.

En las tablas (9.4.8), y (9.4.9.) se muestran los datos estadísticos de los descriptores de forma, así como sus gráficas en las figuras (9.4.5.) y (9.4.6.) respectivamente. Como se puede observar el factor de compacidad nos discrimina a los objetos lineales como el lápiz, la pluma y el marcador; la excentricidad discrimina a los objetos circulares como el sacapuntas y el diurex.

FACTOR DE COMPACIDAD NORMALIZADO				
OBJETO	PROMEDIO	+ DESVEST	- DESVEST	DESVEST
Lápiz	0.692	0.866	0.518	0.174
Pluma	0.792	0.962	0.622	0.170
Sacapuntas	0.118	0.126	0.109	0.008
Diurex	0.100	0.101	0.100	0.001
Borrador	0.148	0.181	0.115	0.033
Calculadora	0.139	0.204	0.074	0.065
Corrector	0.243	0.296	0.190	0.053
Goma	0.121	0.155	0.087	0.034
Resistol	0.119	0.143	0.096	0.023
Marcador	0.346	0.412	0.279	0.067

Tabla 9.4.8: Promedio y desviación estándar para el Factor de compacidad.

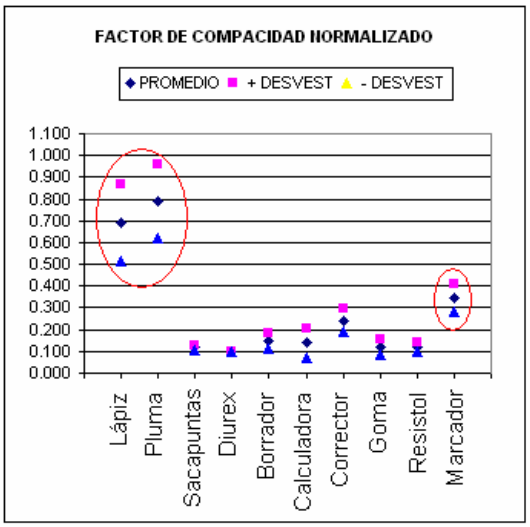


Figura 9.4.5 Gráfica correspondiente al factor de compacidad.

EXCENTRICIDAD				
OBJETO	PROMEDIO	+ DESVEST	- DESVEST	DESVEST
Lápiz	0.998	0.999	0.998	0.000
Pluma	0.999	0.999	0.999	0.000
Sacapuntas	0.258	0.285	0.232	0.026
Diurex	0.159	0.219	0.099	0.060
Borrador	0.919	0.920	0.918	0.001
Calculadora	0.760	0.772	0.748	0.012
Corrector	0.980	0.982	0.978	0.002
Goma	0.745	0.748	0.742	0.003
Resistol	0.730	0.753	0.707	0.023
Marcador	0.993	0.994	0.991	0.001

Tabla 9.4.9: Promedio y desviación estándar para la excentricidad.

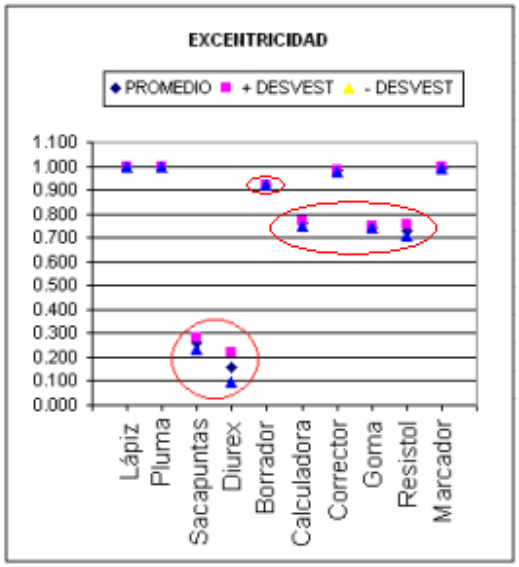


Figura 9.4.6 Gráfica correspondiente a la excentricidad.

El programa # 8 nos presenta la forma de calcular el área, perímetro, excentricidad y centro de masa de un objeto en una imagen binaria. Para el cálculo de la excentricidad se utiliza la función "*regionprops*" de MatLab.

```
clear; close all
imagen = imread('linea.bmp');
nivel = graythresh(imagen);
imagen_byn = im2bw(imagen,nivel);
[ancho largo]=size(imagen_byn);
imagen_byn=double(~imagen_byn)
momento00=0; momento01=0; momento10=0;
    for x=1:1:ancho
        for y=1:1:largo
            momento00=momento00+imagen_byn(x,y);
            momento01=momento01+(x-1)*imagen_byn(x,y);
            momento10=momento10+(y-1)*imagen_byn(x,y);
        end
    end
area=momento00
centrox=momento10/momento00
centroy=momento01/momento00
[ancho largo]=size(imagen_byn);
im=imagen_byn;
perimetro=0;
for x=2:1:ancho-1
    for y=2:1:largo-1
        b=im(x,y-1);d=im(x-1,y);f=im(x+1,y); h=im(x,y+1);
        b=double(b);d=double(d); f=double(f);h=double(h);
        if b==0|d==0|f==0|h==0 perimetro=perimetro+imagen_byn(x,y); end;
    end
end
perimetro
FC=(perimetro^2)/(4*3.1416*area)
propiedades=regionprops(imagen_byn,'all')
excentricidad=propiedades.Eccentricity
per=sum(sum(bwperim(imagen_byn,4)))
```

Programa #8: Cálculo de descriptores de forma

DESCRIPTOR DE TEXTURA UTILIZADO

Los descriptores de textura no son invariantes a la distancia y son muy sensibles a cambios de iluminación. El cálculo de las matrices de coocurrencia tiene un costo computación elevado para imágenes de 256 niveles de gris, la solución que se llevo a cabo fue convertir la imagen de 256 niveles de gris a 8 niveles de gris para facilitar el proceso. Para solucionar el problema la invarianza a diferentes distancias se dividió la energía entre el área del objeto.

De acuerdo con los resultados obtenidos en la tabla (9.4.10), existen pequeñas variaciones de energía a diferentes distancias y rotaciones para el mismo objeto, lo cual dificulta su clasificación.

IMAGEN	ROTACIÓN	DISTANCIA	ENERGIA
 IM0900025	0^0	25 cm	0.304
 IM0900030	0^0	30 cm	0.401
 IM0900035	0^0	35 cm	0.428

Tabla 9.4.10: Descriptor de textura a diferentes distancias y rotación.

De acuerdo a la tabla (9.4.11.) y su gráfica en la figura (9.4.7), el único objeto que contiene mayor energía y que se puede discriminar fácilmente es el borrador con un promedio 0.756, en los otros objetos puede haber confusión, aunque para esta etapa solamente puede haber dos o tres objetos con semejantes características de color y forma de acuerdo a los diez objetos a reconocer.

ENERGIA NORMALIZADA				
OBJETO	PROMEDIO	+ DESVEST	- DESVEST	DESVEST
Lápiz	0.376	0.439	0.313	0.063
Pluma	0.519	0.600	0.438	0.081
Sacapuntas	0.117	0.135	0.099	0.018
Diurex	0.190	0.209	0.171	0.019
Borrador	0.759	0.887	0.631	0.128
Calculadora	0.193	0.273	0.113	0.080
Corrector	0.333	0.376	0.290	0.043
Goma	0.286	0.341	0.231	0.055
Resistol	0.478	0.577	0.379	0.099
Marcador	0.423	0.470	0.376	0.047

Tabla 9.4.11: Promedio y desviación estándar para la energía.

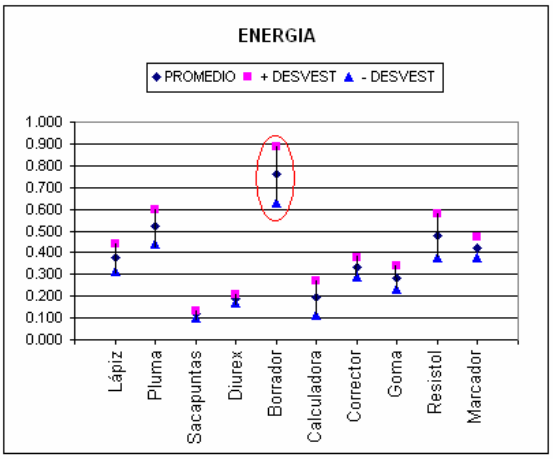


Figura 9.4.7 Gráfica correspondiente a la energía.

El programa #9 nos presenta la forma de calcular la energía de una imagen de 5 columnas, 5 filas y tres niveles de gris para rotaciones de 0, 45 y 90 grados.

```
clear all; close all
im=[0 0 1 1 2; 0 0 1 1 2; 0 0 1 1 2; 0 0 1 1 2; 0 0 1 1 2]
 [filas columnas]=size(im);
zeros
textura0=zeros(4,4);textura45=zeros(4,4); textura90=zeros(4,4);
for fila=1:filas
 for columna=1:columnas
  for x=0:2
   for y=0:2
     if columna~=columnas & im(fila,columna)==x & im(fila,columna+1)==y
        textura0(x+1,y+1)=  textura0(x+1,y+1)+1; end;
     if columna~=columnas & fila~=1 & im(fila,columna)==x & im(fila-1,columna+1)==y
        textura45(x+1,y+1)= textura45(x+1,y+1)+1; end;
     if fila~=1 & im(fila,columna)==x & im(fila-1,columna)==y
        textura90(x+1,y+1)= textura90(x+1,y+1)+1; end;
   end
  end
 end
end
textura0
textura45
textura90
energia0=sum(sum((textura0.^2)))
energia45=sum(sum((textura45.^2)))
energia90=sum(sum((textura90.^2)))
figure, imshow(im);title('textura');
```

Programa #9: Cálculo de los descriptores de textura.

9.5. Resultados del reconocimiento e interpretación

Tomando como base de aprendizaje los descriptores de color, forma y textura del las 240 imágenes para los 10 objetos utilizados en nuestro proyecto; se implemento el clasificador "*K próximos vecinos*" en tres ocasiones, uno para la clasificación por color utilizando tres descriptores (*Rojo,Verde,Azul*), otro para la clasificación por forma en el cuál se utilizaron 2 descriptores (*Factor de compacidad, Excentricidad*) y finalmente otro para la clasificación por textura en el cuál se utilizó un solo descriptor (*Energía*).

CLASIFICACIÓN POR COLOR

De las pruebas realizadas (*ver tabla 9.5.1*), se concluyó que la clasificación por color detectaba correctamente el 86.66 % de las clases de objetos. El lápiz de color blanco se confunde con el corrector y el resistol que son del mismo color. El diurex se confunde con el borrador y el sacapuntas que también son de color amarillo.

Resultados aceptables, aunque se pretende mejorar la clasificación utilizando descriptores de forma y de textura.

CLASIFICACIÓN POR COLOR			
OBJETO	**CANTIDAD DE OBJETOS**	**ACIERTOS**	**ERRORES**
Lápiz	24	18	6
Pluma	24	22	2
Sacapuntas	24	24	0
Diurex	24	18	6
Borrador	24	24	0
Calculadora	24	21	3
Corrector	24	20	4
Goma	24	24	0
Resistol	24	13	11
Marcador	24	24	0
TOTAL	**240**	**208**	**32**
PORCENTAJE		**86.66%**	**13.13%**

Tabla 9.5.1: Resultados de la clasificación por color.

Como se puede observar en la figura (9.5.1.), la clasificación por color utilizando el clasificador "*K Próximos Vecinos*" nos discrimina algunas clases que se encuentran separadas como el marcador, el sacapuntas, borrador y la goma. Los objetos de color blanco se confunden como el resistol, el lápiz, el corrector. Algunos objetos de la misma clase están dispersos o junto con otras clases y es muy difícil su clasificación.

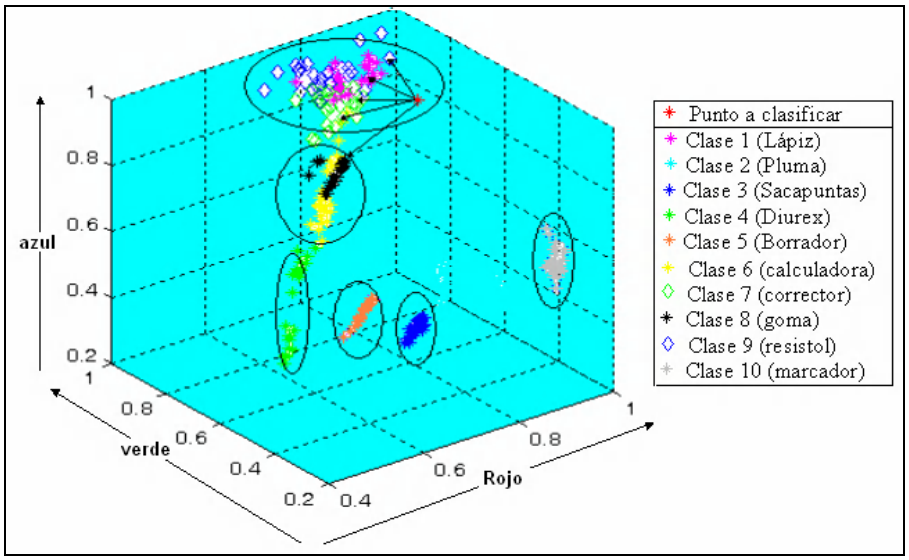
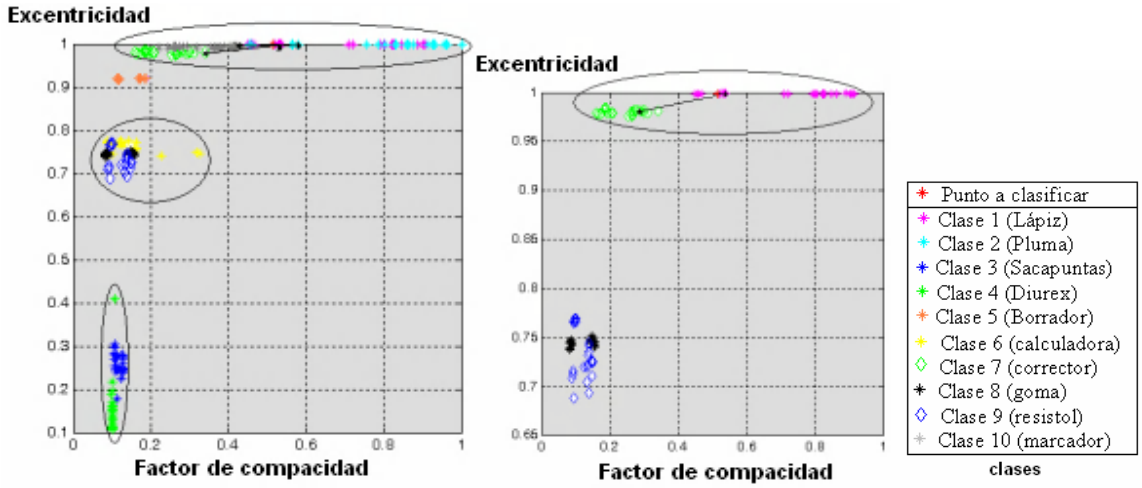


Figura 9.5.1 Dispersión de clases para el color (RGB).

CLASIFICACIÓN POR FORMA

Al utilizar la clasificación por forma y de acuerdo a la gráfica de la figura 9.5.2.a, nos damos cuenta que se sigue teniendo el mismo problema de clasificación con los objetos que tienen forma semejante como el lápiz, la pluma, el marcador y el corrector; pero si utilizamos la clasificación por forma únicamente con las clases en las cuales existe confusión en la clasificación por color, se puede llegar a conocer con certeza la clase a la que pertenece el objeto o solamente se confundiría con otra clase de igual color y forma (*ver figura 9.5.2.b.*).



a) b)

Figura 9.5.2: a) Dispersión de 10 clases de objetos en la clasificación de forma, b) Dispersión de 4 clases de objetos en la clasificación de forma.

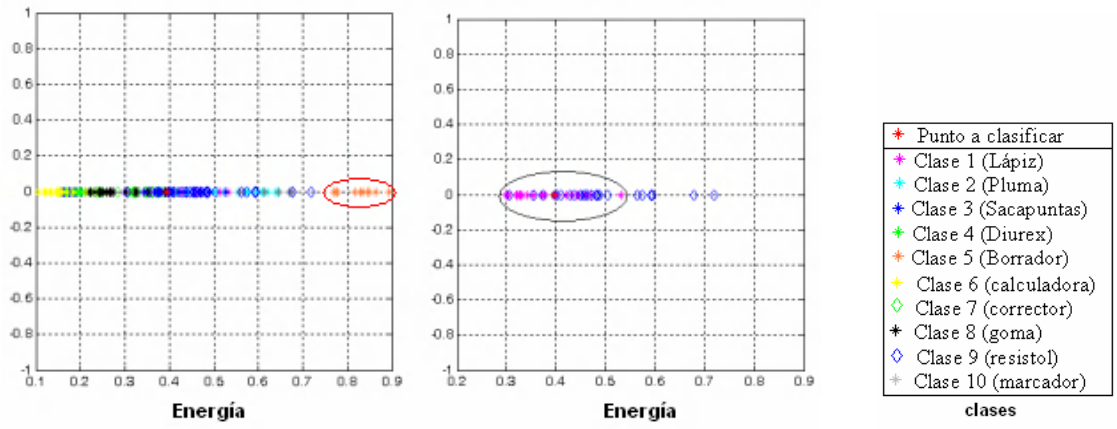
En la tabla 9.5.2. se observa que el sacapuntas, la goma y el marcador ya no aparecen en la clasificación por forma, ya que en la etapa de clasificación por color se obtuvo el reconocimiento del 100% de estos objetos con un porcentaje superior al 90% y la clasificación por forma únicamente se realiza con los objetos que se encuentran por debajo de este porcentaje aunque la clasificación por color sea la correcta.

CLASIFICACIÓN POR FORMA			
OBJETO	**CANTIDAD DE OBJETOS**	**ACIERTOS**	**ERRORES**
Lápiz	24	20	4
Pluma	22	22	0
Sacapuntas	0	0	0
Diurex	24	24	0
Borrador	1	1	0
Calculadora	24	18	6
Corrector	24	24	0
Goma	0	0	0
Resistol	24	23	1
Marcador	0	0	0
TOTAL	**143**	**132**	**11**
	PORCENTAJE	**92.33%**	**7.69%**

Tabla 9.5.2: Resultados de la clasificación por forma.

CLASIFICACIÓN POR TEXTURA

En la clasificación por textura se utilizó un vector con las características de la energía pertenecientes a los 10 objetos (*ver la figura 9.5.3.*). Como se puede observar, en la clasificación por textura únicamente se discrimina a la clase 5 (*borrador*), pero sigue existiendo la confusión entre las demás clases; pero si clasificamos por textura únicamente dos clases con semejante color y forma podemos conocer la clase a la que pertenece el objeto, por ejemplo: al clasificar únicamente la clase 1 (*lápiz*) y la clase 9 (*resistol*) se observan algunos puntos de la clase 9 que se encuentran muy lejanos del punto a clasificar, por lo tanto, el punto a clasificar pertenece a la clase 1.



a) b)

Figuras 9.5.3: a) Dispersión de 10 clases de objetos en la clasificación de textura, b) Dispersión de 4 clases de objetos en la clasificación de textura.

En la clasificación por textura solamente se clasificaron los objetos que no cumplían con el 90% de certeza en la clasificación por color y por textura. Los resultados obtenidos se muestran en la tabla (9.5.3.) y se puede observar que no son muy buenos a pesar de que la clasificación se realiza con dos o tres clases que tienen el mismo color y forma.

CLASIFICACIÓN POR TEXTURA			
OBJETO	**CANTIDAD DE OBJETOS**	**ACIERTOS**	**ERRORES**
Lápiz	10	6	4
Pluma	0	0	0
Sacapuntas	0	0	0
Diurex	0	0	0
Borrador	0	0	0
Calculadora	17	9	8
Corrector	1	1	0
Goma	0	0	0
Resistol	1	0	1
Marcador	0	0	0
TOTAL	**29**	**16**	**13**
PORCENTAJE		**55.17%**	**44.82%**

Tabla 9.5.3: Resultados de la clasificación por textura.

La utilización del clasificador "*KNN*" en la clasificación por color, textura y forma en forma secuencial nos arroja en forma conjunta un total de 13 errores en 240 objetos a identificar y llegando a un porcentaje de aciertos de **94.58%** y combinando los resultados obtenidos en forma individual en cada clasificación únicamente para los objetos que no se tiene un porcentaje superior al 90% en la clasificación por textura, se reducen los errores a 7 de 240, obteniendo así una clasificación total del **97.08%**.

En el programa # 10 se implementa la utilización del clasificador "*K próximos vecinos*" para tres clases, cada una con 100 números aleatorios y densidad normal. Se calculan las distancias del punto a clasificar con cada uno de los puntos de las tres clases y se elige la clase que contenga más puntos cercanos.

```
clear; close all; var1=10; var2=10; var3=10; m1=0; m2=50; m3=100;
x1=input('dame la coordenada x del punto a clasificar? ');
y1=input('dame la coordenada y del punto a clasificar? ');
nc1=0; nc2=0; nc3=0;
c1=randn(2,100)*var1+m1; c2=randn(2,100)*var2+m2; c3=randn(2,100)*var3+m3;
k=round(sqrt(300)); todas=zeros(2,300);
todas(:,1:100)=c1; todas(:,101:200)=c2; todas(:,201:300)=c3;
minimas=zeros(2,13)
for p=1:1:300
     x2=todas(1,p); y2=todas(2,p);
     distancias(p)=abs(sqrt((x2-x1)^2+(y2-y1)^2));
end
for i=1:1:13
     [minvalor pos]=min(distancias)
     minimas(1,i)=minvalor; minimas(2,i)=pos;
     if (pos>0)&(pos<=100)  nc1=nc1+1; end
     if (pos>100)&(pos<=200) nc2=nc2+1; end
     if (pos>200)&(pos<=300) nc3=nc3+1; end
     distancias(1,pos)=max(distancias);
end
numeros(1:3)=[nc1 nc2 nc3]
cuenta=0;
[maxnumero claseganadora]=max(numeros);
for j=1:1:3
    if (numeros(j)==maxnumero) cuenta=cuenta+1;  end
end
if (cuenta>1) claseganadora=4; end %rechazo por ambiguedad
maxnumero
claseganadora
c1x=c1(1,1:100); c1y=c1(2,1:100); c2x=c2(1,1:100); c2y=c2(2,1:100);
c3x=c3(1,1:100); c3y=c3(2,1:100);
figure
hold on
plot (c1x,c1y,'y*',c2x,c2y,'g*',c3x,c3y,'b*',x1,y1,'r*')
```

Programa #10: Clasificador "K próximos vecinos".

9.6. Implementación del cálculo de la distancia del objeto

Realizando la trigonometría, se determinó que el puntero del láser debe de estar a 12.7 cm del centro de visión de la *WebCam* con un ángulo de inclinación de 70 grados para que la distancia del objeto sea de 35 cm. (ver figura 9.6.1).

El ángulo $\beta = 180 - 90 - 70$ = 20

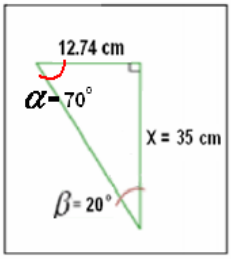


Figura 9.6.1 Cálculo de la distancia

la distancia "*x*" puede calcularse con la fórmula de la ecuación (7.2.1) para la función tangente:

$$\tan\alpha = \frac{12.74}{x} = 0.364$$

despejando el valor de *x:*

$$x = \frac{12.74}{0.364} = 35 \text{ cm}$$

Se puede concluir con los datos anteriores que a una distancia de 35 cm el láser apunta al centro de la imagen de 640 x 480 píxeles (*como se puede apreciar en la figura 9.6.2*). Para la distancia de 30 cm el puntero del láser se encuentra a una distancia aproximada de 34 píxeles con respecto del centro de la imagen y para la distancia de 25 cm el puntero del láser se encuentra a una distancia aproximada de 126 píxeles.

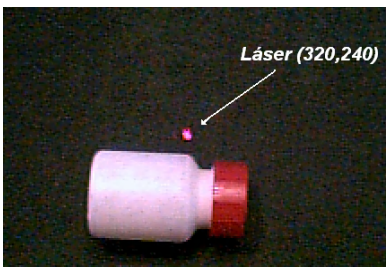


Figura 9.6.2 Coordenadas del láser al centro de la imagen para una distancia de 35 cm.

Para determinar las coordenadas del puntero del láser que se proyecta en la imagen se utiliza la resta de dos imágenes (*previamente convertidas a escalas de gris*); la primera con el puntero del láser y la segunda sin el puntero. La imagen obtenida en teoría debería contener solo el puntero del láser; pero no es así, ya que contiene un poco de ruido que se debe eliminar por medio de un filtro de área, eliminando de esta manera los puntos pequeños (*menores a 30 píxeles*). A continuación se calcula el centro de masa del puntero y de esta manera determinar sus coordenadas (*Ver figuras 9.6.3 a 9.6.7*).

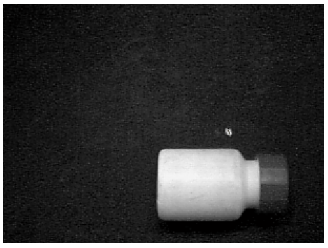


Figura 9.6.3: *Imagen con el puntero del láser*

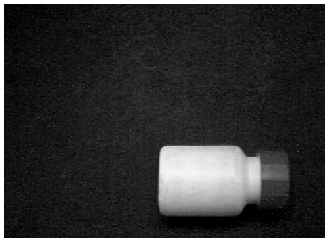


Figura 9.6.4: *Imagen sin puntero del láser*

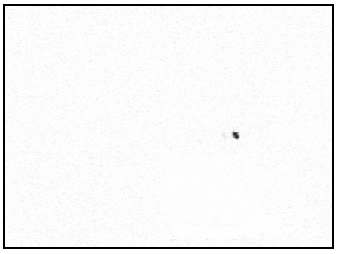


Figura 9.6.5: *Diferencia de imágenes*

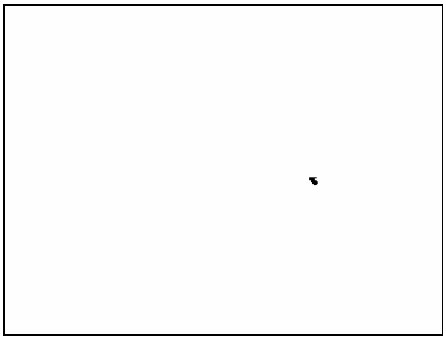


Figura 9.6.6: *Imagen resultado de aplicar un filtro de tamaño.*

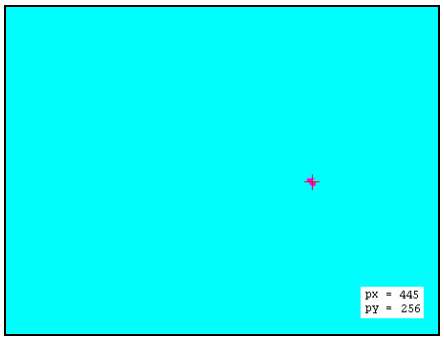


Figura 9.6.7: *Coordenadas del centro de masa.*

Por último se calcula la distancia euclidiana con respecto del centro de la imagen utilizando la siguiente fórmula:

$$distancia = \left| \sqrt{(320 - px)^2 + (240 - py)^2} \right| \qquad (9.6.1)$$

Por ejemplo, si los valores del centro de masa del puntero son: px = 445 y py = 256. Aplicando la fórmula (9.6.1.) obtenemos:

$$distancia = \left| \sqrt{(320 - 445)^2 + (240 - 256)^2} \right| = \left| \sqrt{(15625) + (256)} \right| = \underline{126.019}$$

El programa #11 nos muestra la codificación para la determinación de la distancia del puntero del láser. Se puede observar un margen de error de 10 píxeles en cada distancia a calcular, ya que la calibración del láser o la determinación del centro de masa del puntero pueden variar un poco.

```
clear all; close all
im1_color=imread('imagen0135con.bmp');
rojo1=double(im1_color(:,:,1));
verde1=double(im1_color(:,:,2));
azul1=double(im1_color(:,:,3));
im2_color= imread('imagen0135sin.bmp');
rojo2=double(im2_color(:,:,1));
verde2=double(im2_color(:,:,2));
azul2=double(im2_color(:,:,3));
im_gris1=round(rojo1(:,:)*0.33)+round(verde1(:,:)*0.59)+round(azul1(:,:)*0.11)
im_gris2=round(rojo2(:,:)*0.33)+round(verde2(:,:)*0.59)+round(azul2(:,:)*0.11)

resta=double(im_gris1)-double(im_gris2);
resta=uint8(resta);
[X,map] = gray2ind(resta,2);
im_byn = X;
im_byn = bwareaopen(im_byn,30);

[labeled2,numObjects] = bwlabel(im_byn,4);
graindata = regionprops(labeled2,'basic')
for p=1:1:numObjects
    if graindata(p).Area==max([graindata.Area]) objeto=p; end
end
xy=round(graindata(objeto).Centroid);
px=xy(1)
py=xy(2)

distancia=abs(sqrt((320-px)^2+(240-py)^2))
if (distancia>=0)   && (distancia <= 10) dist='35cm'; end
if (distancia>=30)  && (distancia <= 40) dist='30cm'; end
if (distancia>=120)&& (distancia <= 130) dist='25cm'; end
dist
```

Programa #11: Cálculo de la distancia del objeto.

9.7. Implementación de la grabación de archivos de audio

La grabación se llevó a cabo mediante el programa coolEdit Pro a 44000 Hz en estereo. Se realizo la grabación en archivos WAV con el nombre de cada uno de los objetos a identificar, así como también de las diferentes distancias previamente seleccionadas: 25cm, 30cm y 35cm (*ver figura 9.7.1.*).

Nombre	Tamaño	Tipo
borrador	161 KB	Archivo WAV
calculadora	172 KB	Archivo WAV
corrector	150 KB	Archivo WAV
diurex	135 KB	Archivo WAV
goma	134 KB	Archivo WAV
lapiz	200 KB	Archivo WAV
marcador	152 KB	Archivo WAV

Nombre	Tamaño	Tipo
pluma	136 KB	Archivo WAV
resistol	173 KB	Archivo WAV
sacapuntas	212 KB	Archivo WAV
25cm	328 KB	Archivo WAV
30cm	292 KB	Archivo WAV
35cm	343 KB	Archivo WAV

Figura 9.7.1: Archivos de audio

Para la eliminación de ruido de fondo se aplicó el filtro "*Noise Reduction*" como se puede apreciar en las figuras 9.7.2 y 9.7.3 donde se muestran dos archivos de audio: uno con ruido de fondo y el segundo con la aplicación del filtro "*Noise Reduction*".

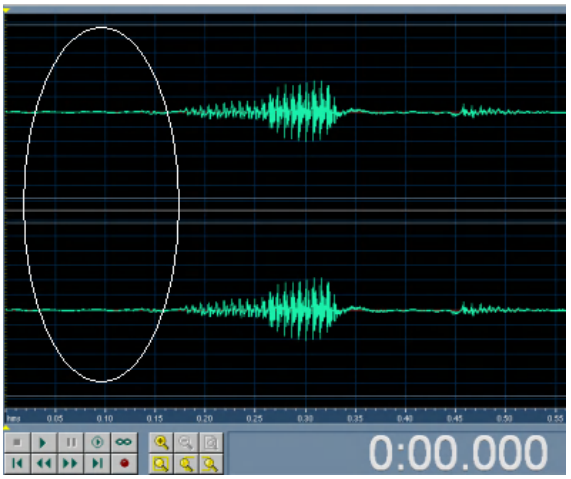


Figura 9.7.2: Archivo con ruido de fondo

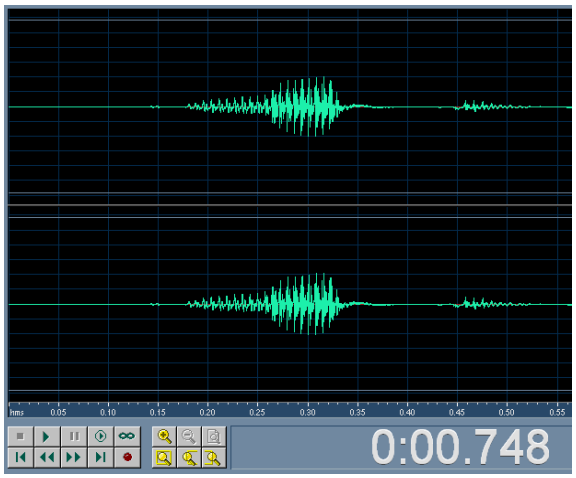


Figura 9.7.3: Archivo resultante de aplicar "Noise Reduction"

La reproducción de los archivos en formato "*wav*" se realizó en *Matlab* mediante las instrucciones *wavread* y *wavplay*. A continuación se presenta su implementación:

```
1    [y,Fs,bits] = wavread('lapiz.wav');
2    wavplay(y,Fs);
```

Conclusiones

Sin duda alguna, la etapa principal del proyecto es la captación de la imagen, la cámara (W*ebCam*) que se utilizó en el presente proyecto, genera ruido en la imagen y variaciones de color; a pesar de lo anterior, en la clasificación por color obtuvimos el 86.66% de efectividad, pero combinada con otros descriptores en cascada se obtuvo el 97.08 %.

Utilizando el método presentado en este proyecto y teniendo una excelente captación de la imagen es posible llegar a una clasificación cercana al 100 % . El método consiste en clasificar en la primera etapa a los objetos por color (*RGB*) y eliminar a objetos que no sean del color del objeto a clasificar. En la segunda etapa se clasifican por forma (*factor de compacidad y excentricidad*) únicamente a los objetos más parecidos en color al objeto en cuestión, esto permite el mejor desempeño del clasificador y finalmente si no se tiene la certeza de la clasificación se utiliza el descriptor de textura (*energía*).

El reconocimiento de objetos en forma automática no es una tarea fácil ya que en la actualidad no existe un método capaz de reconocer cualquier tipo de objeto en condiciones variadas de iluminación ni en el tiempo en que lo realiza un ser humano.

El presente trabajo es uno mas de los cientos de algoritmos que se han desarrollado hasta el momento; pero para la realización de este proyecto implicó la utilización de diferentes técnicas de procesamiento digital de imágenes, la utilización de fórmulas matemáticas, el desarrollo de algunas rutinas de programación, así como la investigación bibliográfica sobre las técnicas que emplearon diferentes autores para el reconocimiento de objetos.

La aportación principal del presente proyecto es la forma de aplicar el clasificador "*K próximos vecinos*", eliminando objetos que no cumplen con las características de clasificación. Otra aportación son los códigos fuente de las rutinas implementadas que pueden ser utilizados por los nuevos estudiantes para futuros proyectos de investigación.

Bibliografía

Bibliografía general

[1] Aguilar, Karla P, "*Probando el desempeño de varias combinaciones de clasificadores y vectores de rasgos en la determinación de la identidad y el numero de objetos circulando en una banda transportadora*", Tesis de maestría, 2001, Ed. Instituto Politécnico Nacional.

[2] Legarda, Arturo S, "*Localización de placas en vehículos automotores*", Tesis de maestría, 2001, Ed. Instituto Tecnológico de Chihuahua.

[3] Rangel, Rafael S, "*Nueva técnica para contar objetos en imágenes*", Tesis de maestría, 2001, Ed. Instituto Politécnico Nacional.

[4] Meléndez, Alba R, "*Estimación de fondo y primer plano en secuencias de imágenes para la detección de objetos en movimiento*", Tesis de maestría, 2003, Ed. Instituto nacional de astrofísica óptica y electrónica de puebla.

[5] Jiménez, Antonio, "*Sistema de reconocimiento y localización de objetos cuasi-esféricos por telemetría láser. Aplicación a la detección automática de frutos para el robot Agribot*", Tesis de doctoral, 2000, Universidad Complutense de Madrid.

[6] Pajares, Gonzalo, "*Visión por computador"*, Alfaomega, México, 2002.

[7] González, Rafael C, "*Tratamiento digital de imágenes"*, Addison-Weslwy, New York, 1996.

[8] Zagal, Juan, "*Adaptación evolutiva de un sistema visual de reconocimiento de objetos para el campeonato de fútbol robótico robocup"*, Universidad de chile, 2002.

[9] Mery, Domingo, "*Detección de fallas en piezas fundidas usando una metodología de reconocimiento de patrones*", Universidad de Santiago de Chile, 2003.

[10] Tadeo, Fernando, "*Clasificación de microorganismos mediante procesado de imagen*", Ingeniería de Sistemas y Automática, 2001

[11] López, Manuel, "*manufactura inteligente utilizando visión para robots",* Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas, 2004.

[12] Alba, José Luís, "*Colorimetría y Captura de Imagen*", Universidad de Vigo, 2006.

[13] Palacios, Rafael , "*Procesamiento de Color*", Universidad Pontificia de Madrid, 2002.

[14] Gutierrez, Violeta, "*Iluminación para las aplicaciones de Visión Artificial"*, Universidad Nacional de Quilmes, Agosto de 2005.

[15] Urdiales, Cristina, "*El color en imagen digital*", 2002.

[16] Pardiñas, Jorge, "*Operaciones Básicas en el dominio espacial*", Universidad Jesuita de Guadalajara, 2002.

[17] Lippincott, Williams*, "Artificial Vision for the Blind by Connecting a Television Camera to the Visual Cortex"*, American Society of Artificial Internal Organs, ASAIO Journal 2000.

[18] Murphy, Erik, "*Shared Features for Scalable Appearance-Based Object Recognition"*, IEEE 2005 Workshop on Applications of Computer Vision, Breckenridge, CO. USA. Jan 5th-7th, 2005.

[19] Shimizu, Akinobu, "*A Modified Exoskeleton and Its Application to Object Representation and Recognition*", IEICE TRANS. INF. & SYST., VOL. E85D, No.5, MAY 2002.

[20] Simard, Patrice, "*A Foreground/Background Separation Algorithm for Image Compression*", 2002.

Bibliografía electrónica

[21] http://www.cochenet.com/sabelotodo/articulos/sistemas%20ayuda/coche_inteligente_vw/coche_inteligente_vw.htm

[22] http://www.elmundo.es/elmundo/2003/03/20/enespecial/1048122765.html

[23] http://neurologia.rediris.es/congreso-1/conferencias/neuropsicologia-2-2.html

[24] http://vision.arc.nasa.gov/

[25] http://www.robocup.org.mx/

[26] http://www.topshareware.com/SoundEdit-Pro-transfer-1450.htm

[27] http://ftp.syntrillium.com/pub/cool_edit/ce2kmain.exe

[28] http://www.laorejadigital.com/default.php?echohtml=%2Fdemos.php3%3F

[29] http://www.utilidades-utiles.com/descargar-sound-forge.html

[30] http://www.qsound.com/

[31] http://www.winamp.com/

[32] http://audacity.sourceforge.net/

[33] http://www.desarrolloweb.com/articulos/1450.php

[34] http://www.f64digital.com/.../content.php?content.21

[35] http://www.quesabesde.com/camdig/articulos.asp

[36] http://www.informaticamedica.org.ar/numero11/sumario.htm