
PONTIFICIA **UNIVERSIDAD CATÓLICA** DEL PERÚ

# PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

## ESCUELA DE GRADUADOS

SISTEMA DE VISIÓN ARTIFICIAL PARA EL RECONOCIMIENTO Y MANIPULACIÓN DE OBJETOS UTILIZANDO UN BRAZO ROBOT

**Tesis para Optar el Grado de:**

**MAGISTER EN INGENIERÍA DE CONTROL Y AUTOMATIZACIÓN**

**Presentado por:**

**EDDIE ANGEL SOBRADO MALPARTIDA**

Lima- Perú
2003

# CONTENIDO

## Capítulo 3

### Descripción y Extracción de características.



## Capítulo 4

### Reconocimiento Automático de Patrones.

# RESUMEN

En este proyecto, un brazo robot permitirá seleccionar objetos (tornillos, tuercas, llaveros, etc) que se encuentran en una mesa, independiente de la posición y orientación. El problema se aborda mediante un esquema de Visión Artificial consistente en 6 etapas: obtención de la imagen, preprocesamiento, segmentación, extracción de características, clasificación y manipulación con el brazo robot.

Se implementa técnicas de aprendizaje y clasificación automáticas para un sistema de visión. La clasificación en si se puede enfrentar de múltiples maneras, generalmente los clasificadores tradicionales, que se basan en métodos estadísticos y/o estructurales y otro esquema como la red neuronal el que presenta algunas ventajas frente a los métodos tradicionales y el cual será realizado.

Una vez reconocida y localizada una pieza determinada, se dará la señal de mando al manipulador robótico para que este lo recoja y lo coloque en una posición determinada previamente por el operador. Por tanto, mediante Visión Artificial y el brazo Robot Scorbot ER-IX, vamos a reconocer y manipular piezas.

# INTRODUCCION

Una de las líneas de investigación y desarrollo muy interesantes, es lo relativo a la "imitación" de nuestros órganos de los sentidos: audición y visión. En ello, no sólo se espera dotar un computador de la capacidad de "percibir" sonidos, sino de reconocer, es decir de "identificar " lo percibido. Se ha avanzado bastante en el reconocimiento de voz, mientras se ha avanzado menos en materia de visión-interpretación.

La visión artificial por computadora es una disciplina en creciente auge con multitud de aplicaciones, como inspección automática, reconocimiento de objetos, mediciones, robótica etc. El futuro es aún más prometedor; la creación de máquinas autónomas capaces de interaccionar inteligentemente con el entorno pasa necesariamente por la capacidad de percibir éste. Estas máquinas, dotadas de computadores cada vez más rápidos y potentes, encuentran una de sus mayores limitaciones en el procesamiento e interpretación de la información suministrada por sensores visuales.

La manipulación de objetos en líneas de ensamble de robots es una tarea donde un gran conjunto de diferentes tipos de objetos aparecen en orientación y posición arbitraria. En este caso el reconocimiento y la aplicación satisfactoria del robot depende crucialmente de las ***técnicas de reconocimiento*** de objetos confiables que se empleen para poder cumplir totalmente ciertos requerimientos importantes.

En este trabajo se implementa un sistema el cual permitirá a un brazo robot seleccionar y manipular objetos conocidos que se encuentran en una mesa, independiente de la posición y orientación. El problema se aborda mediante un esquema de Visión Artificial consistente en 6 etapas: obtención de la imagen, preprocesamiento, segmentación, extracción de características, clasificación y manipulación con el brazo robot.

La *captura de la imagen* es el conjunto de operaciones que se efectúan para transformar la iluminación de una escena en una señal digital. Las imágenes no siempre se presenta en un formato adecuado para su análisis, por lo que el siguiente proceso es el *preprocesamiento* de una imagen, en el cual se utilizan técnicas encaminadas a realizar la mejora de la imagen, como son el nivel de gris, contraste, eliminación del ruido, el realce de algunas características de interés, etc. Una vez que esta imagen esta en condiciones de ser procesada, se tienen que hallar los objetos dentro de la imagen de forma independiente, y esto se hace a través de la *segmentación*, que es un proceso que divide la escena en objetos. Cada uno de los objetos puede ser clasificado, por lo que la siguiente tarea es la *clasificación* o *extracción de características* para el reconocimiento. El *reconocimiento* es la identifiación de cada objeto en la escena mediante una etiqueta.

Con este proyecto se pretende lograr algunos objetivos, tales como dar inicio a la aplicación de visión artificial a la robótica, en donde aparte de reconocer y manipular objetos de orientación y posición desconocida, se pueda posteriormente, implementar el propia sistema de control del robot basado en la retroalimentación visual. Además dicho estudio puede dar inicio a la aplicación de sistemas de visión a otras modalidades de la robótica, por ejemplo robótica móvil.

# Capítulo 1

# Visión Artificial

## 1.1 Introducción.

Podríamos decir que la Visión Artificial (VA) describe la deducción automática de la estructura y propiedades de un mundo tridimensional posiblemente dinámico, bien a partir de una o varias imágenes bidimensionales del mundo. Las imágenes pueden ser monocromáticas (de niveles de gris) o colores, pueden provenir de una o varias cámaras e incluso cada cámara puede estar estacionaria o móvil.

Las estructuras y propiedades del mundo tridimensional que queremos deducir en visión artificial incluyen no sólo sus propiedades geométricas, sino también sus propiedades materiales. Ejemplos de propiedades geométricas son la forma, tamaño y localización de los objetos. Ejemplos de propiedades de los materiales son su color, iluminación, textura y composición. Si el mundo se modifica en el proceso de formación de la imagen, necesitaremos inferir también la naturaleza del cambio, e incluso predecir el futuro.

La entrada a un sistema de VA es una imagen obtenida por un elemento de adquisición, mientras que su salida es una descripción de la escena, la cual ha sido obtenida a partir de la imagen. Por un lado, esta descripción debe estar relacionada de algún modo con aquella realidad que produce la imagen y, por el otro, debe contener toda la información requerida para la tarea de interacción con el medio ambiente que se desea llevar a cabo, por ejemplo mediante un robot. Esto es, la descripción depende en alguna forma de la entrada visual y debe proporcionar información relevante y utilizable por el robot.

## 1.2 Aplicaciones de la Visión Artificial

El amplio espectro de aplicaciones cubierto por la VA, se debe a que permite extraer y analizar información *espectral, espacial y temporal* de los distintos objetos.

La *información espectral* incluye frecuencia (color) e intensidad (tonos de gris). La *información espacial* se refiere a aspectos como forma y posición (una, dos y tres dimensiones). La *información temporal* comprende aspectos estacionarios (presencia y/o ausencia) y dependientes del tiempo (eventos, movimientos, procesos).

La mayoría de las aplicaciones de la visión artificial podemos clasificarlas por el tipo de tarea, entre las que mencionaremos a continuación.

La **medición** o **calibración** se refiere a la correlación cuantitativa con los datos del diseño, asegurando que las mediciones cumplan con las especificaciones del diseño. Por ejemplo, el comprobar que un cable tenga el espesor recomendado.

La **detección de fallas** es un análisis cualitativo que involucra la detección de defectos o artefactos no deseados, con forma desconocida en una posición desconocida. Por ejemplo, encontrar defectos en la pintura de un auto nuevo, o agujeros en hojas de papel.

La **verificación** es el chequeo cualitativo de que una operación de ensamblaje ha sido llevada a cabo correctamente. Por ejemplo, que no falte ninguna tecla en un teclado, o que no falten componentes en un circuito impreso.

El **reconocimiento** involucra la identificación de un objeto con base en descriptores asociados con el objeto. Por ejemplo, la clasificación de cítricos (limones, naranjas, mandarinas, etc.) por color y tamaño.

La **identificación** es el proceso de identificar un objeto por el uso de símbolos en el mismo. Por ejemplo, el código de barras, o códigos de perforaciones empleados para distinguir hule de espuma de asientos automotrices.

El **análisis de localización** es la evaluación de la posición de un objeto. Por ejemplo, determinar la posición donde debe insertarse un circuito integrado.

**Guía** significa proporcionar adaptativamente información posicional de retroalimentación para dirigir una actividad. El ejemplo típico es el uso de un *Sistema de Visión para guiar un brazo Robótico* mientras suelda o manipula partes. Otro ejemplo sería la navegación en vehículos autónomos.

## 1.3 La Visión Artificial en Robótica

El sentido de visión en los robots industriales es el elemento esencial que permite a estos presentar características de adaptabilidad, flexibilidad y capacidad de reorganización.

La visión juega un papel muy importante en los sistemas de **manipulación automática** que merezcan el calificativo de **inteligentes** y en general en los sistemas flexibles de manufactura, pues permite la retroalimentacion sensorial fina que hace ampliar las capacidades de los robots.

La VA proporciona la descripción del estado que guardan los elementos del puesto de trabajo, así como su evolución en el tiempo, información que el sistema de control del robot utiliza en la generación y modificación de sus planes de trabajo, en el monitoreo de la ejecución de tareas y en la detección de errores e imprevistos.

El empleo de sistemas de VA en la manipulación controlada sensorialmente permite, por un lado resolver problemas de conocimiento a priori del ambiente, precisión, costo y fiablilidad y por otro lado, permite a los robots industriales evolucionar en ambientes variables.

En este contexto, un Sistema de VA debe realizar las siguientes funciones:

- Reconocimiento de piezas o conjuntos, así como sus posiciones de equilibrio.
- Determinación de la posición y orientación de piezas con relación a un referencial base.
- Extracción y ubicación de rasgos significativos de las piezas, con objeto de establecer servomecanismos visuales que permitan su manipulación robotizada.
- Inspección en línea y verificación de que el proceso ha sido realizado satisfactoriamente (control de calidad sin contacto).

Por otro lado, las aplicaciones de la VA en los robots controlados sensorialmente, son básicamente los siguientes:

- Manipulación de objetos aislados acarreados por bandas transportadoras: normalmente las piezas tienen una posición de equilibrio única y presentan una proyección fácilmente identificable (aun cuando esté en contacto con otra pieza) mediante técnicas de reconocimiento.
- Manipulación de objetos acomodados aleatoriamente en contenedores: en este caso los objetos presentan una proyección no única, parcialmente oculta, su posición y orientación son aleatorias. Por tanto requiere de una potencia de calculo mayor y de algoritmos de tratamiento de imágenes más sofisticados
- Ensamble: esta tarea se resume en los siguientes 4 puntos: permite identificar piezas, tomarlas y presentarlas en una forma predeterminada para ensamblarlas

con otras piezas; provee retroalimentación visual en el posicionamiento dinámico de las herramientas de ensamble; provee control continuo del órgano terminal sobre la trayectoria deseada; y permite la inspección en línea.

## 1.4 Diagrama de Bloques del Sistema de VA a implementar.

El objetivo del sistema de percepción para el robot, basado en sensores de visión, es el de transformar la imagen del medio ambiente, proporcionada por la cámara, en una descripción de los elementos presentes en el entorno del robot. Dicha descripción deberá contener información necesaria para que el robot efectúe los movimientos que permitirán la ejecución de la tarea programada.

Para alcanzar estos objetivos, se deberá elaborar las siguientes funciones:

a. Iluminación de la escena a capturar mediante la cámara.
b. Captación de la imagen del entorno significativo del robot, su conversión Analógico/Digital y su adquisición por una computadora.
c. Mejoramiento de la imagen y realzado de las características geométricas y topológicas relevantes desde el punto de vista de la aplicación.
d. Segmentación de la imagen.
e. Descripción de la imagen y/o extracción de características.
f. Reconocimiento de los objetos.
g. Elaboración de las consignas para el sistema de control del robot.

El diagrama de bloques del sistema a implementar se muestra en la figura 1.

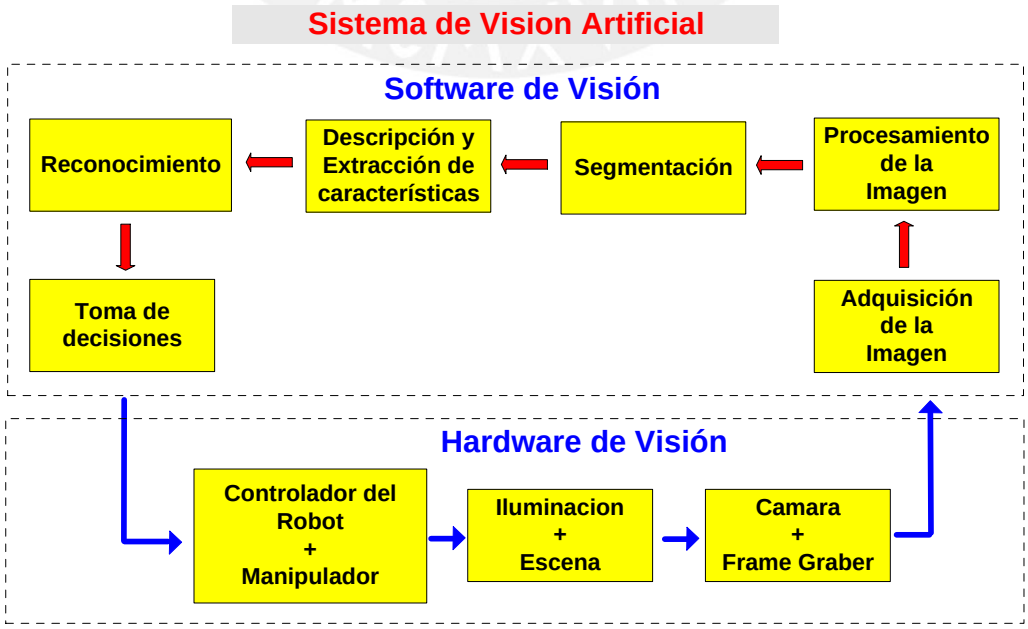


Figura 1.1 Diagrama de Bloques del Sistema de VA a implementar

## 1.5 Hardware y Software utilizado.

Daremos a conocer las herramientas utilizadas, tanto de software como de hardware en el desarrollo de la aplicación.

### 1.5.1 Hardware

El hardware que se utiliza, que tiene como finalidad capturar imágenes digitales de los objetos a reconocer, se detallan a continuación:

a. ***Sistema de iluminación:*** consiste de un fluorescente para iluminar la escena (objetos).

b. ***Cámara de Video:*** corresponde a una cámara de video CCD con formato NTSC (National Television Standard Committee), la cual se utiliza en la obtención de imágenes de los objetos.

c. ***Frame Grabber:*** es una tarjeta de adquisición de imágenes Analógico/Digital que se inserta en un slot de la computadora. Este se usa para obtener un "frame" o cuadro desde una señal de video, con el fin de generar una imagen digitalizada, con una resolución de 8 bits por pixel (256 niveles de gris). La resolución de la imagen capturada es de 240*320 pixels.

d. ***Computadora:*** para el desarrollo y ejecución de los programas se utilizó una computadora personal con un procesador Pentium III a 550 Mhz.

e. ***Brazo manipulador Scorbot ER-IX y su controlador:*** es el elemento que seleccionará las piezas deseadas.

### 1.5.2 Software

El software de visión artificial para reconocimiento de objetos será desarrollado en lenguaje de programación **Visual C**. Sin embargo para validar el desarrollo de los algoritmos implementados en dicho lenguaje, usamos previamente **Matlab**, debido a las facilidades que este lenguaje ofrece para la lectura y procesamiento de imágenes.

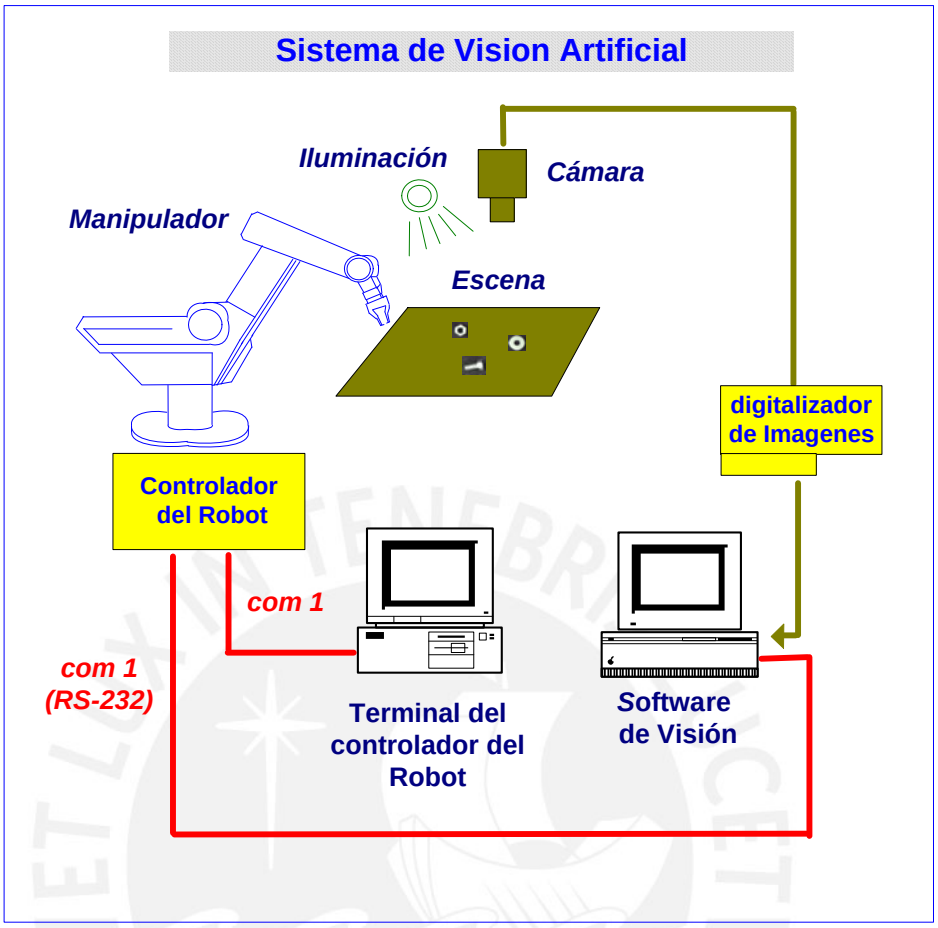



Figura 1.2 Hardware y Software del Sistema de VA a implementar

# Capítulo 2

# Etapas del Sistema de Visión Artificial

## 2.1 Introducción.

En un sistema de visión artificial se incluyen diversas técnicas, tales como el procesamiento de imágenes (captura, transformación, codificación de imágenes) o como el reconocimiento de formas (teoría estadística de decisiones, enfoques sintácticas y neuronales aplicados a la clasificación de patrones). En este tipo de sistemas, además se incluyen técnicas de modelado geométrico y procesos de conocimiento. En vista a. esto, en este capítulo se trata de describir las etapas a considerar en el sistema de visión a implementar en el presente trabajo.

## 2.2 Adquisición de la imagen.

El sistema que permite la captura y/o adquisición de la imagen, está formado por los siguientes elementos:

### 2.2.1 Cámara.

Es el dispositivo encargado de transformar las señales luminosas que aparecen en la escena, en señales analógicas capaces de ser transmitidas por un cable coaxial. Se divide en dos partes, el sensor, que captura las propiedades del objeto en forma de señales luminosas y lo transforma en señales analógicas, y la óptica que se encarga de proyectar los elementos adecuados de la escena ajustando una distancia focal adecuada.

Los sensores de visión usados mas recientemente son los basados en matrices de dispositivos acoplados por carga CCD; estos transductores proporcionan una señal con amplitud proporcional a la luminosidad de la escena y realizan una digitalización espacial completa en dos dimensiones (líneas y columnas), pues descomponen la imagen en una matriz de puntos.

La codificación de la brillantez de cada elemento de imagen o pixel, obtenido de la digitalización espacial, se hace generalmente en 8 bits, mientras que la resolución de la discretización espacial de una imagen puede ser por ejemplo de 320*240 pixeles.

La tecnología CCD *interline transfer* (IT) y *frame interline transfer* (FIT) identifica el tipo de CCD, cada uno de ellos tiene aspectos positivos y negativos. En la práctica el fabricante tiene optimizado el diseño y el tipo de CCD usado raramente determina el funcionamiento completo de la cámara.

- **CCD tipo IT:** la tecnología IT (*interline transfer*) tiene registros separados, protegidos de la luz con una máscara de aluminio opaco ópticamente, como se observa en la figura 2.1. Las cargas proporcionales al contenido de la escena se acumulan en cada elemento del arreglo del sensor. Durante el intervalo vertical, los paquetes de carga son desplazados al arreglo de almacenamiento adyacente. Luego, los elementos del sensor ahora vacíos, capturan el próximo campo mientras la información del arreglo de almacenamiento se transmite fuera para formar la señal de video de salida.

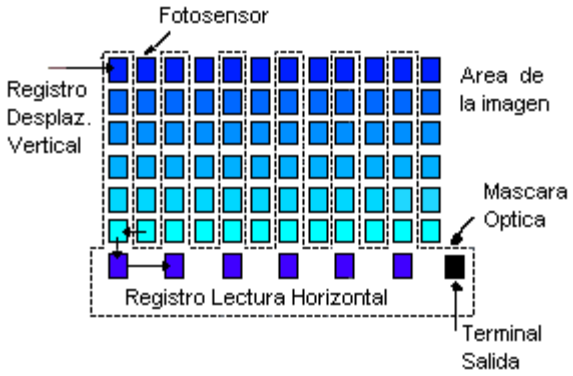



Figura 2.1 CCD tipo IT (*Interline Transfer)*

- **CCD tipo FIT**: la parte superior de este dispositivo opera como un CCD IT. Sin embargo las cargas son rápidamente desplazadas desde el registro de almacenamiento *interline* al registro de almacenamiento protegido totalmente. Los paquetes de carga son mantenidos en el registro *interline* solo por un corto tiempo

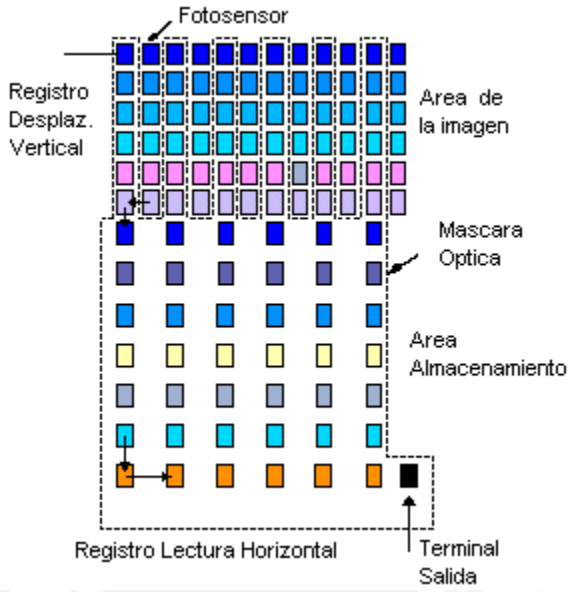



Figura 2.2 CCD tipo FIT (Frame *Interline Transfer)*

### 2.2.2 Digitalizador.

El digitalizador (frame grabber), es el encargado de transformar la señal de vídeo, cualquiera que sea el formato utilizado (NTSC, PAL), en una señal digital capaz de ser capturada, almacenada en memoria y procesada por una computadora. Las principales características de las tarjetas digitalizadoras son precio, controlabilidad, resolución, velocidad y almacenaje, en el sentido de sí los algoritmos de visión pueden tener acceso rápido y fácilmente a los datos.

Existen tarjetas que proporcionan sus propios buffers de memoria y otras que utilizan la memoria del ordenador (vía DMA). Muchas de ellas permiten un preprocesamiento previo de las imágenes, donde el número de tareas implementadas en hardware es muy variable.

La resolución de las tarjetas digitalizadoras y la de las cámaras (sensor) no tiene porque coincidir. Por lo tanto, es importante (sobre todo cuando se emplean técnicas de medición) saber que ocurre con los puntos que faltan o sobran.

### 2.2.3 Sistema de iluminación.

La iluminación de la escena juega un papel crucial en el desarrollo de un sistema visual. Antes de intentar corregir un problema de iluminación por medio de algoritmos muy complicados, es mejor prestar atención e implantar un sistema de iluminación adecuado, para que la captura de la imagen sea correcta. Es mejor un buen sistema de iluminación, que intentar corregir ese problema por software, pues la velocidad de procesamiento será mayor con algoritmos más sencillos.

Por tanto, prestaremos una especial atención a los diferentes sistemas de iluminación, exponiendo una breve descripción de alguno de ellos.

**a. Retroiluminación Difusa.**

Es la más adecuada, si para el reconocimiento o medida de una pieza solo se necesita el contorno y es posible apoyar dicha pieza sobre una superficie transparente. Consiste en iluminar contra la cámara, dejando el objeto entre la cámara y la lámpara. Esta técnica proporciona imágenes con un alto contraste entre la pieza y el fondo, resultando fácilmente segmentable mediante una simple binarización aunque se pierden los detalles de la escena.

Las principales aplicaciones donde se comporta bien esta técnica de iluminación son para medir el grado de porosidad de ciertas sustancias y en inspección dimensional para calcular el tamaño de una pieza.

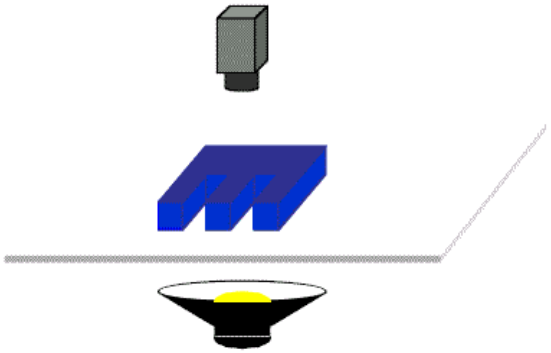


Figura 2.3 Iluminación Difusa

**b. Iluminación Frontal**

Es la más usada, y consiste en iluminar frontalmente la pieza. Presenta más problemas para obtener un buen contraste entre la pieza y el fondo, debido a la aparición de brillos y sombras que alteran las propiedades de las piezas a estudio.

Se emplea en piezas poco reflectoras para evitar los brillos que son bastante molestos, usándose una iluminación difusa muy estudiada para piezas muy reflectoras

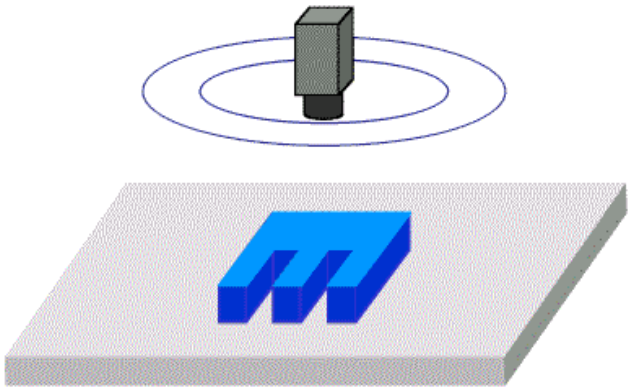


Figura 2.4 Iluminación Frontal

**c. Luz Direccional**

Consiste en una iluminación direccionada en algún sentido en el espacio para destacar una característica concreta del objeto. La principal virtud es la creación de sombras sobre el objeto, lo que puede ayudar a aumentar el contraste de partes tridimensionales y obtener la consiguiente información 3D.

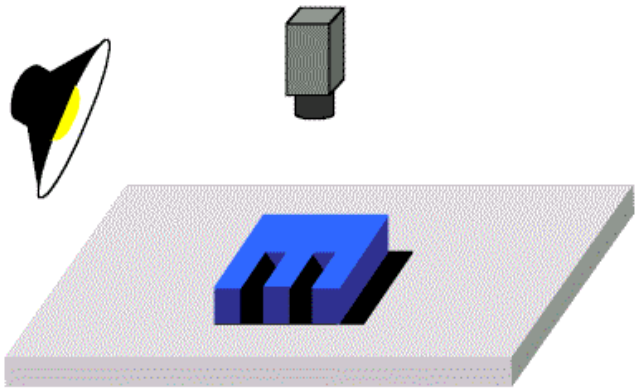


Figura 2.5 Iluminación Direccional

**d. Luz Estructurada**

Consiste en proyectar sobre la pieza unos patrones de luz conocidos modulados (proyección de puntos, franjas o rejillas sobre la superficie de trabajo) y observando la luz reflejada, que también viene modulada, obtener información sobre la estructura

de la superficie del objeto, la cual puede ser reconstruida mediante triangulación. Las fuentes de luz empleadas deben de ser especiales pues deben ser capaces de emitir luz estructurada y suelen ser láseres. Se usa para reconstrucciones 3D de objetos y para conocer su forma.

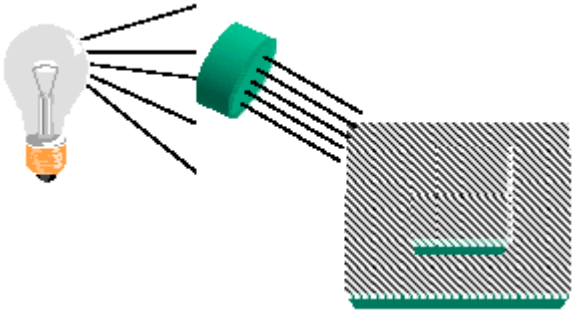


Figura 2.6 Iluminación Estructurada

### 2.2.4 Fuentes de iluminación.

Por otro lado debemos prestar atención también a las diferentes fuentes de iluminación.

**a. Lámparas incandescentes.**

Es la fuente de iluminación más común y consiste en un filamento de tungsteno o halógeno-tungsteno. Como ventaja tiene que existe gran variedad de potencias y como desventaja, que reduce su luminosidad con el tiempo, lo que puede provocar problemas en algunos sistemas de visión.

**b. Tubos fluorescentes.**

Más eficaces que las lámparas y suministran una luz más difusa, que es bueno para piezas muy reflectoras. Existe una gran variedad, tanto en forma (circulares, lineales), como en tamaño con lo que son ampliamente utilizados.

**c. Fibra óptica.**

Para iluminar zonas de difícil acceso o extremadamente pequeñas. Proporciona iluminación constante.

**d. Lásers.**

Empleados para una iluminación con luz estructurada, ya que el láser es capaz de emitir luz estructurada con un control adecuado. Tiene el inconveniente de presentar un mal comportamiento frente a superficies que absorben luz.

## 2.3 Preprocesamiento de la Imagen.

Una etapa importante de la VA es el preprocesamiento de imágenes, es decir, la transformación de la imagen original en otra imagen en la cual hayan sido eliminados los problemas de ruido granular de cuantización o de iluminación espacialmente variable. La utilización de estas técnicas permite el mejoramiento de las imágenes digitales adquiridas de acuerdo a los objetivos planteados en el sistema de VA.

A continuación sólo se mencionara las técnicas de preprocesamiento empleado en el presente trabajo.

### 2.3.1 Conversión a Escala de Grises

En esta parte se trata la conversión de una imagen en color a escala de grises, el equivalente a la luminancia de la imagen. Como sabemos el ojo percibe distintas intesidades de luz en función del color que se observe, esto es debido a la respuesta del ojo al espectro visible la cual se puede observar en la figura 2.7, por esa razón el cálculo de la escala de grises o luminancia de la imagen debe realizarse como una media ponderada de las distintas componentes de color de cada pixel.

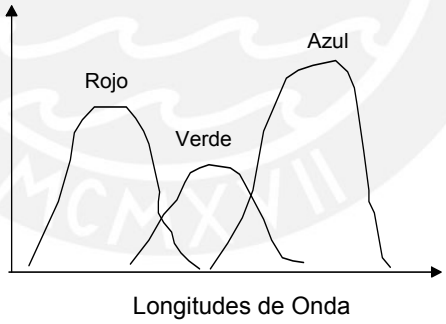



Figura 2.7 Componentes RGB de una imagen

La ecuación de la luminancia es la expresión matemática de ese fenómeno, y los factores de ponderación de cada componente de color nos indican la sensibilidad del ojo humano a las frecuencias del espectro cercanas al rojo, verde y azul.

$$Y = R * 0.3 + G * 0.5 + B * 0.11 \qquad (2.1)$$

Por tanto, para realizar esta conversión basta con aplicar la ecuación 2.1 a cada pixel de la imagen de color, entonces resultará una nueva matriz de un byte por pixel que daría la información de luminancia.

### 2.3.2 Filtro espacial

El empleo de máscaras espaciales para el procesamiento de las imágenes, se denomina frecuentemente *filtrado espacial,* y las propias máscaras se denominan *filtros espaciale*s. Dentro del filtrado espacial, existen los filtros suavizantes, que se emplean para hacer que la imagen aparezca algo borrosa y también para reducir el ruido.

**a. Filtro pasa bajo.**

Para un filtro espacial de 3x3 (grado 3), la construcción más simple consiste en una máscara en la que todos los coeficientes sean iguales a 1. Sin embargo, la respuesta, en este caso es, la suma de los niveles de gris de los nueve pixeles, lo que hace que el resultado quede fuera del rango válido de gris [0,255]. La solución consiste en cambiar la escala de la suma, dividiéndola por el grado de la máscara al cuadrado, en este caso por 9. La figura 2.8 muestra la máscara resultante.

$$\frac{1}{9} * \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Figura 2.8 Filtro espacial de pasa bajo.

**b. Filtro de mediana**

Una de las principales dificultades del método de suavizamiento, es que afecta los bordes y otros detalles de realce. Cuando el objetivo es reducir el ruido, el empleo de los filtros de mediana representa una posibilidad alternativa. En este caso, el nivel de gris de cada pixel se reemplaza por la mediana de los niveles de gris en un entorno de este pixel, en lugar del promedio, como lo hace el filtro pasa bajo.

Este método es particularmente efectivo cuando el patrón de ruido consiste en componentes fuertes y de forma puntiaguda, y la característica que se desea preservar es la agudeza de los bordes.

La mediana *m* de un conjunto de valores es tal que la mitad de los valores del conjunto quedan por debajo de m y la otra mitad por encima. Con el fin de realizar el

filtro de mediana, en el entorno de un pixel, primero se deben extraer los valores del pixel y de su entorno, determinar la mediana y asignar este valor al pixel. Por ejemplo, para un entorno de 3x3, con los valores que se observan en la figura 2.9, se realizan los siguientes pasos:

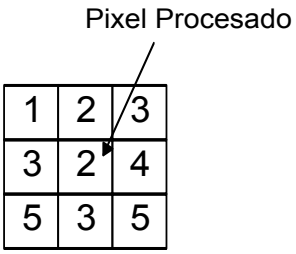



Figura 2.9 Entorno de 3x3.

Se almacenan los valores en un vector:

X[1] = 1, X[2] = 2, X[3] = 3, X[4] = 3, X[5] = 2, X[6] =4, X[7] = 5, X[8] = 3, X[9] = 5

Se hace un ordenamiento en el vector, por valor de nivel de gris:

X[1] = 1, X[2] = 2, X[5] = 2, X[3] = 3, X[4] = 3, X[8] = 3, X[6] =4, X[7] = 5, X[9] = 5

Entonces, el valor de la mediana corresponde a la posición 4, con el valor 3.

Suponiendo que una imagen esta afectada por un cierto error por pixel (debido a ruido en la cámara, ruido shot, etc.), se puede aplicar un operador de proximidad por media de 5x5 para conseguir una mejora notable de la imagen de entrada. A diferencia de un operador de aproximación normal, el filtro de media no causa aplanamiento en los bordes, sino que se limita a eliminar pixeles únicos aleatorios.

## 2.4 Segmentación

La segmentación es el proceso mediante el cual una imagen se descompone en regiones o elementos que pueden corresponder a objetos o parte de objetos. El proceso de segmentación se encarga de evaluar si cada pixel de la imagen pertenece o no al objeto

de interés. Este técnica de procesamiento de imágenes idealmente genera una imagen binaria, donde los pixeles que pertenecen al objeto se representa con un 1, mientras que los que no pertenecen al mismo se representan con un 0. Este tipo de particionamiento esta basado en el análisis de alguna característica de la imagen, tal como los niveles de gris o la textura. A continuación describiremos el método de segmentación basado en la umbralización.

### 2.4.1 Segmentación basado en Pixeles

Este método de segmentación toma en cuenta sólo el valor de gris de un pixel, para decidir si el mismo pertenece o no al objeto de interés. Para ello, se debe encontrar el rango de valores de gris que caracterizan dicho objeto, lo que requiere entonces la búsqueda y el análisis del histograma de la imagen.

El objetivo de este método, es el de encontrar de una manera óptima los valores característicos de la imagen que establecen la separación del objeto de interés, con respecto a las regiones que no pertenecen al mismo; debido a esta característica y si los valores de gris del objeto y del resto de la imagen difieren claramente, entonces el histograma mostrará una distribución bimodal, con dos máximos distintos, lo que debiera generar, la existencia de una zona del histograma ubicada entre los dos máximos, que no presenten los valores característicos, y que idealmente fuera igual a cero, con lo cual se logrará una separación perfecta entre el objeto y la región de la imagen que lo circunda, al establecer un valor umbral ubicado en esta región del histograma. Por lo tanto cada pixel de la imagen, es asignado a una de dos categorías, dependiendo si el valor umbral es excedido o no.

Si el valor del histograma ubicado entre los dos máximos, es distinto de cero, las funciones de probabilidad de los valores de gris del objeto y de la región restante, se solaparán, de tal manera que algunos pixeles del objeto deberán ser tomados como pertenecientes a la región circundante y viceversa. Conocida la distribución de la función de probabilidad de los pixeles del objeto y de la región circundante, es posible aplicar análisis estadístico en el proceso de buscar un umbral óptimo, con el número mínimo de correspondencias erróneas. Estas distribuciones pueden ser estimadas por histogramas locales, los cuales solamente incluyen las regiones correspondientes de la imagen.

En imágenes industriales tenemos generalmente un alto contraste lo cual permite aplicar segmentaciones muy simples como el mencionado, para distinguir dos clases de regiones en la imagen: región objeto y región fondo.

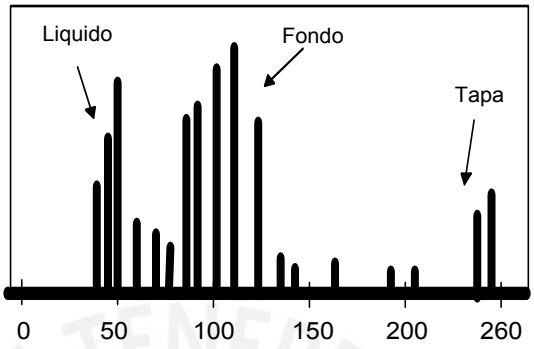



Figura 2.10 Histograma de la imagen de botella

Figura 2.11 Segmentación de la imagen de botella.

En la imagen de la figura 2.10, el histograma nos indica que el fondo (gris claro) es lo que ocupa más espacio en la imagen, porque es el que contiene mayor número de pixeles en la escala de grises 125 (aprox.). Luego tenemos el líquido (refresco, color negro) y la tapa y partes de la etiqueta (color blanco). Si aplicamos un umbral (threshold) a esta imagen, a partir del límite que establece el líquido (aproximadamente nivel 50 de la escala de grises), todo lo demás se convierte en color negro (0), y el líquido se convierte en color blanco (1). De esta forma, la imagen pasa a ser binaria, y el liquido queda claramente separado de todo lo demás que aparece en la imagen

### 2.4.2 Operaciones Morfológicas

Las operaciones morfológicas son métodos para procesar imágenes binarias basado sobre formas. Estas operaciones toman una imagen binaria como entrada y dan como resultado una imagen binaria como salida. El valor de cada pixel en la imagen de salida esta basado sobre el correspondiente pixel de entrada y sus vecinos. Dentro de las operaciones morfológicas tenemos la dilatación y erosión, las cuales serán tratadas a continuación.

La dilatación adiciona pixeles a los limites del objeto (es decir los cambia de off a on), y la erosión remueve pixeles sobre los limites del objeto (los cambia de on a off).

**a. Dilatación**

Sea A y B conjuntos de $Z^2$ y $\varnothing$ representando al conjunto vacío, la dilatación de A por B, se representa $A \oplus B$, en la ecuación (2.2):

$$A \oplus B = \left\{ x / (\hat{B})_x \cap A \neq \Phi \right\} \tag{2.2}$$

Por tanto, el proceso de dilatación consiste en obtener la reflexión de B sobre su origen, después, cambiar esta reflexión por x. La dilatación de A por B es entonces, el conjunto de todos los desplazamientos x, tales que $\hat{B}$ y A se solapen en al menos un elemento distinto de cero. Basándose en esta interpretación, la ecuación (2.2) se puede volver a representar mediante la ecuación (2.3):

$$A \oplus B = \left\{ x / \left[ (\hat{B})_x \cap A \right] \subseteq A \right\} \tag{2.3}$$

Al conjunto B, se le conoce normalmente como el *elemento de estructura* de la dilatación. Como vemos, se toma el elemento de estructura B como una máscara de convolución. Aunque la dilatación se basa en operaciones de conjunto, mientras que la convolución se basa en operaciones aritméticas, el proceso básico de mover a B respecto a su origen, y desplazarlo después sucesivamente de tal forma que se deslice sobre el conjunto (imagen) A, es análogo al proceso de convolución expuesto anteriormente.

Los componentes del conjunto B (elemento estructurador), pueden ser ceros o unos. La cantidad de ceros y/o unos determina, en conjunto con el tamaño del elemento estructurador, el efecto que produce su utilización en las operaciones morfológicas.

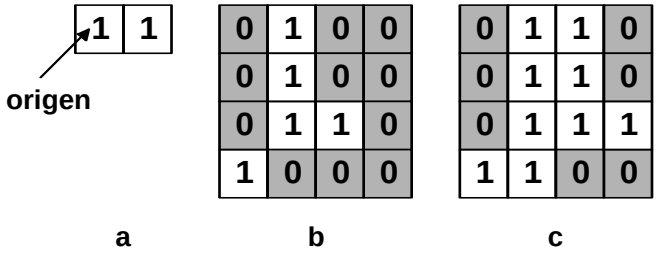



Figura 2.12 Ejemplo de dilatación. (a) Elemento estructural *B* (b) Imagen *A*. (c) Resultado de la dilatación *A* ⊕B

Al usar el elemento estructurador que se muestra en la figura 2.12a, y si se tiene como imagen de entrada la figura 2.12b y después de aplicar el proceso de dilatación, se obtiene la imagen de salida ilustrada en la figura 2.12c.

El estado de cualquier pixel dado en la imagen de salida es determinado aplicando una regla a los vecinos del correspondiente pixel en la imagen de entrada. La siguiente regla define la operación para la dilatación: "Para dilatación, si cualquier pixel en los vecinos del pixel de entrada es on, el pixel de salida es on. De otra manera el pixel de salida es off."

**b. Erosión**

Para los conjuntos A y B de $Z^2$, la erosión de A por B representada por A Θ B, se define como la ecuación (2.3):

$$A\Theta B = \{x/(B)_x \subseteq A\} \qquad (2.4)$$

que dice, que la erosión de A por B es el conjunto de todos los puntos x tal que B, trasladado por x, está contenido en A. Como en el caso de la dilatación, la ecuación (2.4) no es la única definición de la erosión.

Sin embargo, esta ecuación, normalmente, es más adecuada en implementaciones prácticas de la morfología. Si se toma como ejemplo el elemento estructurador que se muestra en la figura 2.13a, y la imagen de entrada de la figura 2.13b, se obtiene la imagen de salida que se muestra en la figura 2.13c, en la cual el proceso de erosión ha sido aplicado.

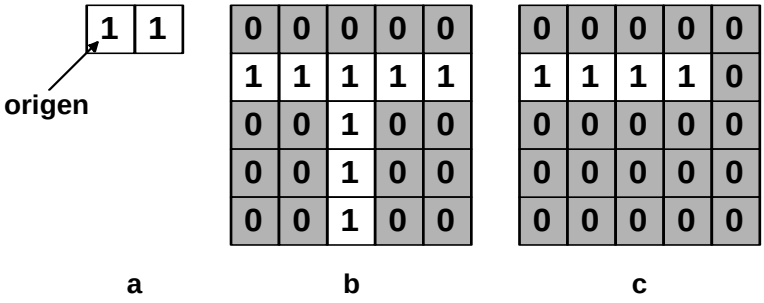



Figura 2.13 (a) Elemento estructural (b) Imagen original, (c) Imagen erosionada.

El estado de cualquier pixel dado en la imagen de salida es determinado aplicando una regla a los vecinos del correspondiente pixel en la imagen de entrada. La siguiente regla define la operación para la erosión: "Para erosión, si cada pixel en el vecino del pixel de entrada esta on, el pixel de salida estará on. De otra manera el pixel de salida esta off."

### 2.4.3 Etiquetado y Cuenta de objetos

En la práctica llegar a la imagen binaria no suele ser suficiente para realizar una descripción adecuada, por lo que el proceso de segmentación se prolonga aplicando diversas técnicas sobre este tipo de imágenes.

Existe una gran cantidad de técnicas de análisis para imágenes binarias, con propósitos tan variados, entre estas técnicas está el contar, etiquetar objetos y filtrado de objetos según su tamaño.

Una de las operaciones más comunes en visión es encontrar las componentes conectadas dentro de una imagen. En la figura 2.14 vemos un ejemplo de una imagen y su imagen de componentes conectadas, en donde se ha etiquetado con un numero a las componentes conectadas.

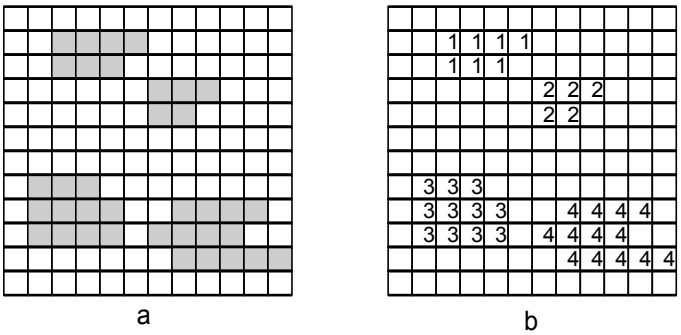



Figura 2.14 (a)Imagen (b)Etiquetado y cuenta de objetos

Se ha de disponer de una definición consistente de conectividad para demarcar todos los objetos en imágenes binarias y ser capaces de idear algoritmos para etiquetarlos y contarlos.

El etiquetado es una técnica que, partiendo de una imagen binaria, nos permite etiquetar cada uno de los objetos conectados presentes en la imagen. Esto va a posibilitar:

- Distinguir los objetos respecto del fondo (propiedad intrínseca de la imagen binaria).
- Distinguir un objeto respecto de los demás objetos.
- Conocer el número de objetos en la imagen.

En general, los algoritmos para etiquetar dan buenos resultados con objetos convexos, pero presentan problemas cuando aparecen objetos que tienen concavidades (formas en U), como se observa en la figura 2.15 donde diferentes partes de un mismo objeto pueden acabar con etiquetas distintas, incluso pueden aparecer colisiones de etiquetas.

En este sentido el peor caso que puede plantearse es un objeto con forma de espiral.

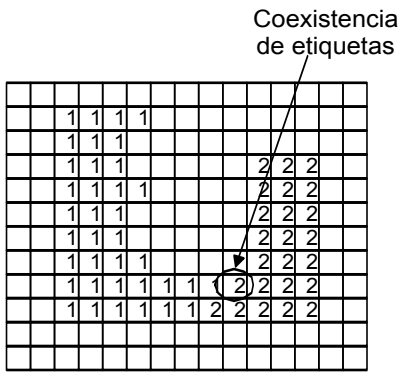



Figura 2.15 Imagen en forma de U etiquetada mediante el algoritmo de etiquetado

### 2.4.4 Filtro de tamaño

El filtrado de tamaño en imágenes binarias esta orientado, principalmente con dos objetivos:

- Eliminación de ruido: debido al proceso de umbralización pueden aparecer regiones en la imagen binaria que son consecuencia del ruido. Tales regiones son normalmente pequeñas.
- Reducir el número de objetos de la imagen (hay objetos de un determinado tamaño que no interesan para la aplicación): En algunas ocasiones resulta interesante eliminar de una imagen binaria aquellos objetos que no superen un tamaño determinado, o por el contrario, eliminar los objetos que si superen este tamaño.

Los objetos son filtrados teniendo en cuenta su tamaño o área o utilizando las operaciones de erosión y dilatación (o expansión) como se analizó anteriormente.

Sin embargo, en muchas aplicaciones es conocido que los objetos de interés son de un tamaño mayor que T pixeles (área). Todas las componentes de la imagen que tienen un tamaño igual o menor que T pixeles se eliminan; para ello se cambian los correspondientes pixeles al valor del fondo (0).

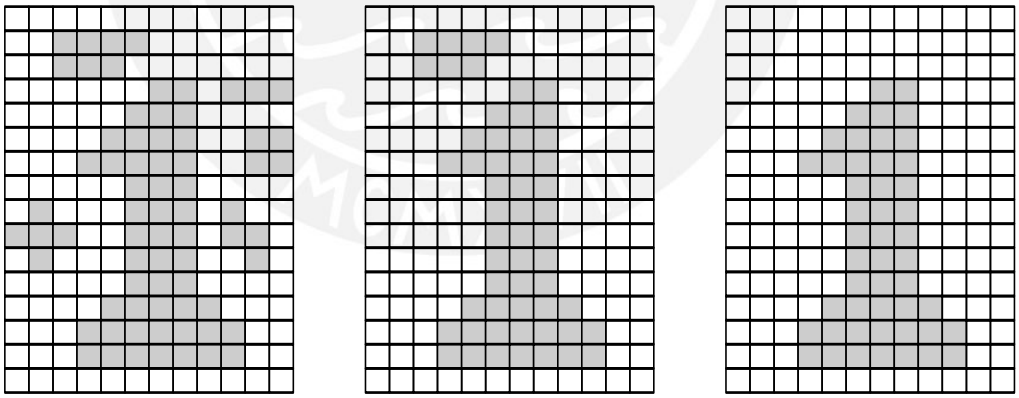



Figura 2.16 (a)Imagen binaria (b)Filtrado de área t=6 (c)Filtrado de área t=10

## 2.5 Descripción

Con el objeto de poder analizar una imagen será necesario tener una descripción en la memoria de la computadora. Esta descripción no es otra cosa que un modelo que representa las características de la imagen de relevancia para los fines del Sistema de Visión Artificial específicos. Los modelos que describen una imagen pueden ser estadísticas, estructurales, etc pero todos ellos se obtienen a partir de imágenes previamente segmentadas y analizadas.

En el tipo de imágenes utilizados en robótica la información relativa se encuentra en el contorno, entonces los modelos utilizados para describir este tipo de imágenes representaran tan solo contornos presentes en la imagen, y entre los más usuales tenemos a las ***cadenas de códigos***. Este punto se explicará mas detalladamente en el próximo capitulo.

## 2.6 Reconocimiento

Finalmente, una vez que se han realizado las etapas de preprocesamiento, segmentación y extracción de características, se procede ha realizar el reconocimiento.

La fase de reconocimiento consistirá en la clasificación de los objetos presentes en la imagen de acuerdo a sus modelos respectivos. Este punto se tratará mas detalladamente en el próximo capitulo debido a su importancia en el desarrollo del sistema propuesto.

## 2.7 Toma de decisiones y comunicación con el Robot

Una vez que han sido reconocidos los objetos presentes en el campo visual, el sistema de VA deberá calcular las consignas de movimiento que deberán ser enviadas al robot, con el objetivo de que este realice la tarea de manipulación previamente programada.

Un aspecto importante en la interacción del sistema de visión con el robot consiste en la comunicación, así como en la programación de ambos sistemas.

# Capítulo 3

# Descripción y Extracción de Características

## 3.1 Introducción

Para reconocer un objeto es necesario tener una *descripción* de los mismos (descriptor o modelo del objeto). Los descriptores deben de ser independientes del tamaño, localización u orientación del objeto, y deben ser suficientes para discriminar objetos entre sí. Los descriptores se basan en la evaluación de alguna *característica* del objeto, por ejemplo:

*a.* *Descriptores unidimensionales* : códigos de cadena, perímetro, forma del perímetro.

*b.* *Descriptores bidimensionales* : área, momentos de inercia, etc.

*c.* *Descriptores específicos* : número de agujeros, área de agujeros, posición relativa de agujeros, rasgos diferenciadores de un objeto, etc.

Uno de los principales problemas en el reconocimiento de patrones, es encontrar una manera óptima de representar la información original que describe a cada uno de los patrones basado en los *descriptores* mencionados inicialmente. Este problema es conocido como ***extracción de características***. Este proceso de extracción de características trata de reducir la cantidad de información (reducción de dimensionalidad) que representa a cada uno de los patrones, obteniendo de esta forma, un ***vector de características*** que represente de la mejor manera posible al patrón original.

## 3.2 Requisitos en la Extracción de Características

Conviene destacar las propiedades más importantes que deben barajarse en la elección o extracción de las características son:

a. ***Discriminación***: valores numéricos diferentes para objetos de clases diferentes.
b. ***Fiabilidad:*** cambios numéricos pequeños para objetos de una misma clase, es decir los objetos de una misma clase deberán representar la menor dispersión.
c. ***Incorrelación:*** la independencia de las características equivale al principio de la parsimonia, decir lo mismo con la máxima economía de términos. Nunca debe utilizarse características que dependan fuertemente entre sí, ya que no añaden información. Se ha de tener la máxima información con el mínimo número de características.
d. ***Cálculo en Tiempo Real:*** este es un requisito que puede llegar a ser determinante en ciertas aplicaciones de tiempo real, ya que las características deben calcularse en un tiempo aceptable.
e. ***Invarianza:*** frente a transformaciones geométricas como rotación, traslación, escalado.
f. ***Dimensionalidad:*** el tamaño del vector de características debe de ser menor que la del patrón original. Las características deben representar una codificación óptima de la entrada, perdiendo la información que no sea muy importante, se debe reflejar lo esencial del objeto no lo accesorio.

Especialmente, cuando se realiza el reconocimiento de patrones utilizando redes neuronales, la extracción de características debe tratar de obtener un vector de características con una dimensionalidad mucho menor a la del patrón original, puesto que un vector con una dimensionalidad más pequeña que sea dado como entrada a la red neuronal tiene varios beneficios. En primer lugar, la cantidad de pesos que deben de ser aprendidos es menor, y en segundo lugar, al tener menos pesos, el tiempo de entrenamiento puede ser reducido considerablemente.

## 3.3 Extracción de características

El término *"extracción de características*" tiene un doble significado. Por un lado, se refiere al proceso de extraer algunas medidas numéricas e información *relevante* de los datos en bruto de los patrones suministrada por los sensores (representación inicial), para la clasificación. Por otro lado, se define también como el proceso de formar un conjunto

de características (de dimensión n) partiendo de los datos de entrada (de dimensión m>n).

De forma general este problema puede plantearse como sigue. Dado un conjunto de patrones *n*-dimensionales:

$$X = [x_1 \quad x_2 \quad .... \quad x_n]^T \tag{3.1}$$

se trata de obtener un nuevo conjunto (características) *d*-dimensionales

$$Y = [y_1 \quad y_2 \quad .... \quad y_d]^T \quad d \leq n \tag{3.2}$$

Este objetivo puede abordarse de dos formas:

a. **Reduciendo la dimensionalidad de los datos**. Si los patrones son de alta dimensionalidad, el coste computacional asociado a la clasificación puede ser muy alto. Como veremos, muchos clasificadores están basados en cálculos de distancias y estos cálculos pueden depender de forma cuadrática respecto a la dimensionalidad de los patrones. Como otra consideración computacional hay que considerar el espacio de almacenamiento adicional que supone guardar los valores de nuevas variables. Además, algunas de las variables pueden ser redundantes con otras y no aportar información adicional.

   La técnica dedicadas a seleccionar las variables más relevantes se denomina ***selección de características*** y reducen la dimensionalidad de los patrones. Este proceso puede esquematizarse como se indica en la figura 3.1a, en el que un módulo selector recibe patrones *n*-dimensionales (en el ejemplo, *n = 100*) y proporciona como resultado las *d* variables más significativas (en el ejemplo, *d=3*) de acuerdo a algún criterio a optimizar. En este caso *d<n* y el conjunto de las *d* variables seleccionadas es un subconjunto del conjunto original de variables.

b. **Cambiando el espacio de representación**. El objetivo es obtener una nueva representación de los patrones en la que los agrupamientos aparezcan bien separados si son de diferente clase y que haya un agrupamiento por clase. Esto puede conseguirse aplicando alguna transformación sobre los datos originales. Estas

transformaciones suelen ser *transformaciones lineales* y el objetivo suele ser maximizar la varianza.

Estas técnicas reciben el nombre de ***extracción de características*** y producen un nuevo conjunto de variables. Este proceso puede esquematizarse como se indica en la figura 3.1.b, en el que un módulo extractor recibe patrones *n*-dimensionales (en el ejemplo, *n=100*) y proporciona como resultado nuevos patrones *n*-dimensionales de acuerdo a algún criterio a optimizar. Es posible que las nuevas variables estén implícitamente ordenadas, por lo que proporcionan, adicionalmente un procedimiento de selección. En este caso *d=n* y las variables seleccionadas *no* forman un subconjunto del conjunto original de variables.

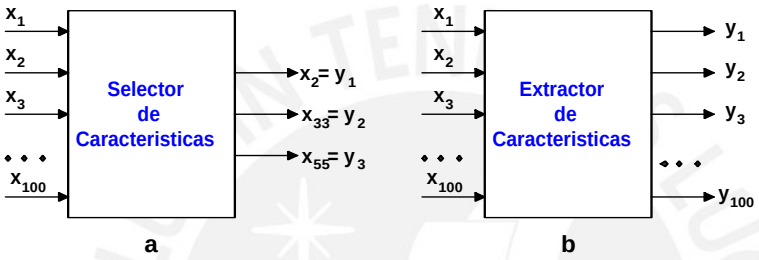



Figure 3.1 (a) Selección de las 3 variables más significativas entre un conjunto de 100. (b) Transformación de las 100 variables originales

Los vectores resultantes del proceso de ***selección*** o ***extracción*** se denominan ***características***, aunque nosotros utilizaremos el término patrón a no ser que queramos resaltar explícitamente que se ha producido un proceso de selección/extracción de características

## 3.4 Métodos de Descripción

Existen muchas formas de describir los objetos presentes en una imagen digital. Los métodos mas utilizados se mencionan a continuación.

### 3.4.1 Descriptores de *F*orma.

Parten de una información binaria de pertenencia de un pixel al objeto. En este tipo tenemos los descriptores siguientes:

a. **Descriptores de Contorno:** Información binaria de pertenencia al contorno: código de cadenas, signaturas, aproximaciones poligonales, representación polar, esqueletización, descriptores de Fourier.
b. **Descriptores de Región:** Información binaria de pertenencia al interior del objeto, no sólo con los del contorno: momentos (área, centro de gravedad, momentos invariantes), esqueletos, descriptores topológicos.

Una imagen binaria generalmente es representada como una matriz, la cual contiene información tanto del objeto como del fondo. Una representación más compacta de la imagen, se puede obtener al almacenar en una estructura de datos, la información de los pixeles que acotan al objeto. De esta forma, bastaría con realizar análisis u operaciones sobre la estructura de datos, para realizar la extracción de los parámetros que definen el objeto, o para aplicar operaciones de modificación de la forma del mismo. Las técnicas que se basan en la utilización de estructuras de datos, para almacenar los puntos que acotan al objeto, son las *técnicas de codificación de contornos*

#### 3.4.1.1 Código de Cadena

Una de las técnicas utilizadas en la extracción de contornos de imágenes a las cuales se les ha aplicado un procedimiento de segmentación, es la del seguimiento de contornos para su codificación. Un algoritmo de seguimiento de contornos examina todas las direcciones establecidas por el **código de cadena** (figura 3.2a).

Comentario [ES1]:

La codificación del cotorno en código de cadena, resultará en una estructura de datos que representa la frontera de una imagen binaria sobre un mallado discreto, en cuyos campos se indica la dirección de los pixeles de la frontera. El primer paso es el de escoger sobre el contorno de la imagen, el pixel inicial cuyas coordenadas deben ser almacenadas en la estructura; para luego seguir el contorno en la dirección de las agujas del reloj. Si el contorno es seguido con ayuda de una vecindad de 4, habrán cuatro posibles direcciones que seguir, mientras que si es seguido con una vecindad de 8 existirán ocho direcciones, como muestra la figura 3.2b. Al utilizar este código, el contorno puede entonces ser codificado con 2 o 3 bits. Los códigos de cadena representan de una manera compacta las imágenes binarias.

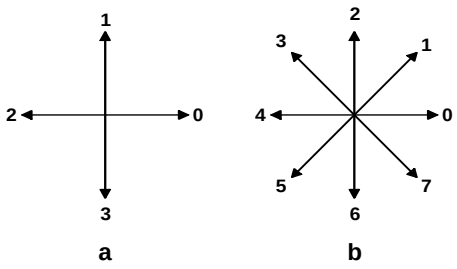



Figura 3.2 Código de cadena. (a)Vecindad de 4 (b)Vecindad de 8.

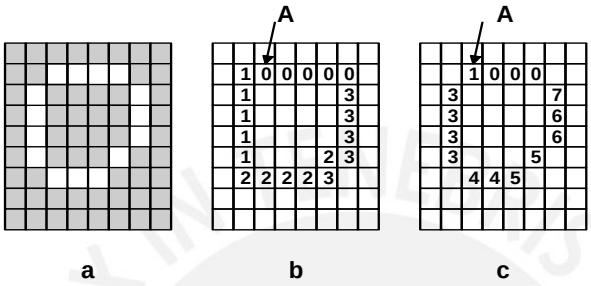



Figura 3.3 Codificaciones de una frontera. (a) Frontera,
(b) 4-conectividad: A00000333323222211111
(c) 8-conectividad: A100076655443333

Una vez que se ha revisado como realizar la representación de objetos (figura 3.3), la idea fundamental, es la estudiar ciertos parámetros que puedan dar una representación adecuada de la forma del objeto.

#### 3.4.1.2 Area

En una imagen digital, el área de un objeto esta dada por el número de pixeles que representan al mismo, por lo tanto el cálculo del área se realiza contando el número de pixeles.

Si una imagen esta representada por un contorno codificado en código de cadena, el cálculo del área se realiza por un algoritmo que trabaja en forma similar a la integración numérica, el cual lo veremos posteriormente.

#### 3.4.1.3. Perímetro

Es un parámetro geométrico que al igual que el área, puede ser calculado a partir del código de cadena. Para realizar este cálculo, es necesario contar la longitud del código, y tomar en consideración que los pasos en direcciones diagonales deben ser multiplicados por un factor igual a raíz cuadrada de dos. El perímetro p de un contorno codificado en código de cadena se expresa como:

$$p = n_e + n_0\sqrt{2} \tag{3.3}$$

donde: $n_e$ representa el número pasos pares del código y $n_o$ el número pasos impares del código.

#### 3.4.1.4. Circularidad

Es uno de los parámetros geométricos utilizados en la comparación de objetos que son observados desde diferentes distancias, ya que el mismo no depende del tamaño del objeto. La circularidad es un número adimensional definido como:

$$C=p^2/A \tag{3.4}$$

Donde p es el perímetro y A es el área. La invariante área y perímetro simplemente determina el área y el perímetro de la imagen y realiza la siguiente relación:

$$\text{Invariante}=\text{Area}/\text{perimetro}^2 \tag{3.5}$$

#### 3.4.1.5. Momentos Generales

La teoría de los momentos proporciona una interesante y útil alternativa para la representación de formas de objetos. Si tenemos un objeto en una región que viene dado por los puntos en los que $f(x,y) > 0$, definimos el momento de orden p,q como:

$$m_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q f(x,y)dxdy \tag{3.6}$$

El interés de estos ***momentos generales*** desde el punto de vista de la caracterización discriminante de los contornos de los objetos es que, estos contornos pueden modelarse como un tipo especial de funciones f(x,y) acotadas y por ende se pueden calcular los momentos generales.

En este sentido, y aquí radica su aplicabilidad en el reconocimiento de formas, dada una función acotada f(x,y) existe un conjunto de momentos generales y viceversa. Es

decir, dado un conjunto de momentos generales se puede reconstruir una función f(x,y) única, simbólicamente:

$$f(x,y) \leftrightarrow \{m_{pq}\} \quad p,q = 0,1\ldots\ldots\infty \tag{3.7}$$

Obviamente esta correspondencia biunívoca no representaría ningún interés práctico, a menos que se pudiera reducir el número de momentos generales a una cantidad manejable.

Particularizando este enfoque basado en la descripción de una función acotada f(x,y) mediante un conjunto finito de sus momentos generales al problema del reconocimiento automático de los contornos (es decir formas cerradas) de los objetos en una imagen digital, es preciso pasar de una integral doble a un doble sumatorio, puesto que la función f(x,y) es ahora una función I(x,y) acotada que toma valores distintos de cero únicamente en el contorno del objeto y en su interior.

Los momentos generales discretos de la función $I_o(x,y)$ serán pues:

$$m_{pq} = \sum_{x=0}^{N-1}\sum_{x=0}^{N-1} x^p y^q I_o(x,y) \qquad p,q = 0,1\ldots\ldots\infty \tag{3.8}$$

dependiendo de cómo se defina $I_o(x,y)$ se tendrá cuatro posibles momentos generales:

$$I_o(x,y) = 1 \text{ contorno}$$
$$I_o(x,y) = 0 \text{ resto} \tag{3.9}$$

$$I_o(x,y) = I(x,y) \text{ contorno}$$
$$I_o(x,y) = 0 \text{ resto} \tag{3.10}$$

$$I_o(x,y) = 1 \text{ objeto}$$
$$I_o(x,y) = 0 \text{ resto} \tag{3.11}$$

$$I_o(x,y) = I(x,y) \text{ objeto}$$
$$I_o(x,y) = 0 \quad \text{resto} \tag{3.12}$$

es decir se puede definir una función acotada $I_o(x,y)$ que tome los valores (1,0) o bien valores (I(x,y),0). Siendo I(x,y) el nivel de intensidad luminosa en la escala de grises. Igualmente se puede establecer el campo de existencia de la función $I_o(x,y)$ bien como el contorno del objeto o bien como el contorno mas su interior.

Los momentos generales que se obtienen a partir de su contorno, aunque lo caracterizan adecuadamente, sufren el grave defecto de ser muy sensibles al ruido y a pequeñas variaciones en la forma de un contorno (siendo preciso tener un mayor

número de momentos generales para robustecer la descripción del contorno). Por el contrario, los momentos generales basados en el contorno mas su interior presentan una mayor robustez.

Por otro lado, para caracterizar la forma de un objeto no es necesario manejar funciones $I_o(x,y)$ multivaluadas, siendo suficiente que tomen un valor único, digamos la unidad.

En consecuencia, para la obtención de características discriminantes de un contorno cerrado es habitual trabajar con la tercera opción, es decir con los momentos generales basados en la función $I_o(x,y)$:

$$I_o(x,y) = 1 \text{ objeto}$$
$$I_o(x,y) = 0 \text{ resto} \qquad (3.13)$$

Un momento de gran interés es el de orden cero (el orden de un momento viene dado por la suma de los índices p y q):

$$m_{00} = \sum_{x=0}^{N-1}\sum_{x=0}^{N-1} I_o(x,y) \qquad (3.14)$$

que obviamente coincide con el ***área del objeto***. El área física se puede calcular sin mas que multiplicar $m_{00}$ por el área física de un pixel, lo cual obliga a un calibrado previo de la cámara. Afortunadamente, desde el punto de vista del reconocimiento de objetos veremos que no es necesario pasar a las dimensiones físicas del objeto, ya que es posible obtener momentos invariantes a traslaciones, giros y tamaños relativos de los objetos.

Los momentos de orden uno, $m_{01}$ y $m_{10}$ junto con $m_{00}$ determinan el llamado ***centro de gravedad del objeto***:

$$\bar{x} = \frac{m_{10}}{m_{00}} = \frac{\sum\sum x I(x,y)}{\sum\sum I(x,y)} \qquad (3.15)$$

$$\bar{y} = \frac{m_{01}}{m_{00}} = \frac{\sum\sum y I(x,y)}{\sum\sum I(x,y)} \qquad (3.16)$$

### 3.4.2 Descriptores de Textura

**a. Niveles de gris**

Parten del histograma del objeto.

**b. Distribución espacial de niveles**

Parten de información de distribución espacial del nivel de gris.

Puesto a que estos descriptores no son usados en la presenta tesis, entonces no se desarrollara en este capitulo.

## 3.5 Momentos Invariantes.

El inconveniente de los momentos generales reside en el hecho de que varía con la posición del objeto dentro del plano de la imagen, así como con el tamaño relativo del objeto, que depende de la distancia entre la cámara y el objeto.

Los cambios en la posición del objeto se deben exclusivamente a las operaciones de giro o traslación. En cuanto a los cambios de tamaños se puede modelar como una operación de homotecia. Por tanto, vamos a transformar sucesivamente los momentos generales para hacerlos invariante a giros y homotecias.

### 3.5.1 Momentos invariante a traslaciones

Los momentos generales se pueden hacer invariantes a traslaciones en el plano sin mas que referirlos al centro de gravedad (x,y), obteniéndose los llamados ***momentos centrales***:

$$u_{pq} = \sum_{x=0}^{N-1}\sum_{x=0}^{N-1}(x-\bar{x})^p(y-\bar{y})^q I_o(x,y) \qquad (3.17)$$

Puesto que el centro de masas $(\bar{x},\bar{y})$ ocupa siempre la misma posición relativa respecto a todos los puntos del objeto, los momentos centrales no varían ante traslaciones de los objetos. El cálculo directo de los momentos centrales es posible en función de los momentos generales:

$$u_{pq} = \sum_{r=0}^{p}\sum_{s=0}^{q}\binom{p}{r}\binom{q}{s}(-\bar{x})^r(-\bar{y})^s m_{p-r,q-s} \qquad (3.18)$$

Así aplicando esta expresión se obtendría los siguientes momentos centrales:

$$u_{00} = m_{00} \tag{3.19}$$

$$u_{01} = u_{10} = 0 \tag{3.20}$$

$$u_{11} = m_{11} - \frac{m_{10}m_{01}}{m_{00}} \tag{3. 21}$$

$$u_{20} = m_{20} - \frac{m_{10}^2}{m_{00}} \tag{3.22}$$

$$u_{02} = m_{02} - \frac{m_{01}^2}{m_{00}} \tag{3.23}$$

$$u_{03} = m_{03} - \frac{3m_{01}m_{02}}{m_{00}} + \frac{2m_{01}^3}{m_{00}^2} \tag{3.24}$$

$$u_{30} = m_{30} - \frac{3m_{10}m_{20}}{m_{00}} + \frac{2m_{10}^3}{m_{00}^2} \tag{3.25}$$

$$u_{21} = m_{21} - \frac{m_{01}m_{20}}{m_{00}} - \frac{2m_{10}m_{11}}{m_{00}} + \frac{2m_{10}^2m_{01}}{m_{00}^2} \tag{3.26}$$

$$u_{12} = m_{12} - \frac{m_{10}m_{02}}{m_{00}} - \frac{2m_{01}m_{11}}{m_{00}} + \frac{2m_{01}^2m_{10}}{m_{00}^2} \tag{3.27}$$

la deducción de estas formulas se desarrollan en el apéndice B.

### 3.5.2 Momentos Invariante a Homotecias

Aunque no se obtengan unos momentos estrictamente invariantes ante cambios del tamaño relativo de un objeto, es decir invariantes a homotecias, suele emplearse la siguiente normalización, en donde los momentos centrales normalizados de orden p+q se define como:

$$n_{pq} = \frac{u_{pq}}{u_{00}} \tag{3.28}$$

donde

$$\gamma = \frac{p+q+2}{2} \quad \text{para } p+q = 2,3....\infty \tag{3.29}$$

Como ya dijimos anteriormente, el problema práctico fundamental es determinar un numero mínimo, pero suficiente, de momentos invariantes que permitan caracterizar cada uno de los objetos del universo de trabajo del sistema de reconocimiento automático.

Para ser rigurosos, habría que evaluar el grado de reconstrucción que se obtendría de un conjunto finito y truncado de momentos. Pero aparte de ser tedioso, este proceso de reconstrucción depende de la forma o contorno de cada objeto. Cuando más irregular sea este, en general se necesitara un mayor numero de momentos para su reconstrucción.

Aunque no se ha dicho, es preciso subrayar que en la obtención del subconjunto de momentos invariantes discriminantes se comienza siempre desde los momentos de menor a mayor orden. Esto es porque es en los momentos de menor orden donde están concentrados las principales características de las formas de un contorno cerrado. Los momentos de orden superior contribuyen en menor grado a la caracterización de la forma.

### 3.5.3 Momentos invariante a Traslaciones, Rotaciones y Homotecias

De los momentos de segundo y tercer orden, pueden derivarse 7 de los llamados ***"momentos invariantes"*** que, no dependen del tamaño ni la posición del objeto, pudiendo ser usados para **la identificación de objetos**. El usuario puede decidir cuales de estos momentos invariantes tendrán mayor significación en su propia aplicación. El siguiente conjunto de momentos invariantes se puede obtener usando únicamente los momentos centrales normalizados de orden 2 y 3:

$$\phi_1 = n_{20} + n_{02} \tag{3.30}$$

$$\phi_2 = (n_{20} - n_{02})^2 + 4n_{11}^2 \tag{3.31}$$

$$\phi_3 = (n_{30} - 3n_{12})^2 + (3n_{21} - n_{03})^2 \tag{3.32}$$

$$\phi_4 = (n_{30} + 3n_{12})^2 + (n_{21} + n_{03})^2 \tag{3.33}$$

$$\phi_5 = (n_{30} - 3n_{12})(n_{30} + n_{12})\left[(n_{30} + n_{12})^2 - 3(n_{21} + n_{03})^2\right] + (3n_{21} - n_{03})(n_{21} + n_{03})\left[3(n_{30} + n_{12})^2 - (n_{21} + n_{03})^2\right] \tag{3.34}$$

$$\phi_6 = (n_{20} - n_{02})\left[(n_{30} + n_{12})^2 - (n_{21} + n_{03})^2\right] + 4n_{11}(n_{30} + n_{12})(n_{21} + n_{03}) \tag{3.35}$$

$$\phi_7 = (3n_{21} - n_{03})(n_{30} + n_{12})\left[(n_{30} + n_{12})^2 - 3(n_{21} + n_{03})^2\right] + (3n_{12} - n_{30})(n_{21} + n_{03})\left[3(n_{30} + n_{12})^2 - (n_{21} + n_{03})^2\right] \tag{3.36}$$

Este conjunto de momentos resulta ser ***invariante a la traslación, la rotación y el cambio de escala***. Otra ventaja de los momentos es su gran facilidad de cálculo, sobre todo aplicándolos a imágenes binarias, ya que pueden ser calculados ópticamente a velocidades elevadas.

## 3.6 Cálculo de los Momentos Generales a partir del código de Cadena

Una vez planteado y desarrollado el fundamento teórico de la aplicación de los momentos de una función bidimensional acotada $I_o(x,y)$, vamos a ver como realizar su cálculo a partir, precisamente del **código de cadena** del contorno de $I_o(x,y)$.

El teorema de Green declara que si una función $f(x,y)$ es integrable sobre una región $A$ y puede ser descompuesta como la suma de las derivadas de dos funciones, es decir:

$$f(x,y) = \frac{\partial Q(x,y)}{\partial x} - \frac{\partial P(x,y)}{\partial y} \tag{3.37}$$

entonces tenemos:

$$\iint_A f(x,y)dxdy = \int_C P(x,y)dx + Q(x,y)dy \tag{3.38}$$

donde $C$ es una curva cerrada, que consiste de muchas pequeñas curvas finitas (figura 3.4). En nuestro caso la función $f(x,y)$ es el momento central $x^p y^q$.

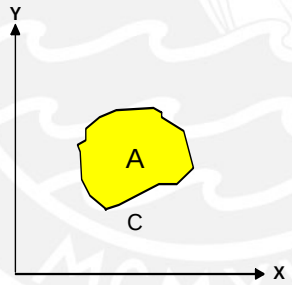



Figura 3.4 Región continua y acotada

La región $A$ esta limitada por la curva $C$, tal como se representa en la figura 3.4; en donde el convenio de los ejes coordenados cartesianos (que en la figura difieren del habitual en imágenes digitales) no influyen en el resultado del teorema y que establece la siguiente igualdad:

$$\iint_A \left(\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y}\right)dxdy = \int_C Pdx + Qdy \qquad (3.39)$$

es decir, el teorema permite pasar de una integral doble extendida sobre la región A, a una integral curvilínea sobre una curva cerrada C. Este resultado nos va permitir calcular los momentos generales, que están definidos mediante integrales dobles, como integrales curvilíneas extendidas sobre el contorno de un objeto, es decir utilizando ***el código de cadena*** que es una aproximación poligonal del contorno.

Si en la expresión (3.33) hacemos:

$$\begin{aligned} P &= -x^p y^{q+1} \\ Q &= x^{p+1} y^q \end{aligned} \qquad (3.40)$$

entonces:

$$\begin{aligned} \frac{\partial P}{\partial y} &= -(q+1)x^p y^q \\ \frac{\partial Q}{\partial x} &= (p+1)x^p y^q \end{aligned} \qquad (3.41)$$

y por tanto:

$$\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y} = (p+q+2)x^p y^q \qquad (3.42)$$

sustituyendo (3.36) en (3.33) y recordando la definición de momento general de orden p+q dada por la expresión (3.4) podemos escribir:

$$m_{pq} = \iint_A x^p y^q dxdy = \frac{1}{p+q+2}\int_C -x^p y^{q+1}dx + x^{p+1}y^q dy \qquad (3.43)$$

en donde obviamente la función $I_o(x,y)$, asociada al objeto vale la unidad en el propio objeto (es decir, en la región A ocupada por el) y cero en el resto del plano.

La integral curvilínea de la expresión (3.37) puede calcularse en función del código de cadena del contorno del objeto. Para eso consideramos un tramo del contorno de un objeto dentro de una imagen digital:

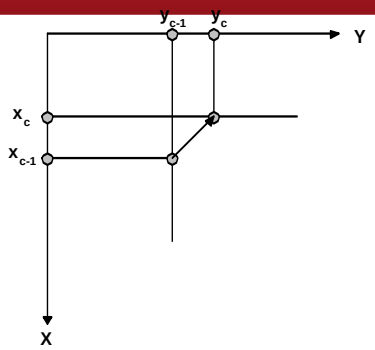



Figura 3.5 Tramo de un pixel

La ecuación de una recta genérica que une dos pixeles consecutivos del contorno es:

$$y - y_c = \frac{dy_c}{dx_c}(x - x_c) = m_c(x - x_c) \tag{3.44}$$

siendo

$$dy_c = y_c - y_{c-1} \tag{3.45}$$
$$dx_c = x_c - x_{c-1} \tag{3.46}$$

en donde $(x_c,y_c)$ y $(x_{c-1},y_{c-1})$ son las coordenadas de dos pixeles consecutivos del contorno. Despejando de (3.44):

$$y = y_c + m_c(x - x_c) \tag{3.47}$$

permite expresar la integral curvilínea de (3.43), que da el momento general de orden p+q en función de x exclusivamente. Para ver esto vamos a calcular el momento general de orden cero $m_{00}$. Así:

$$m_{00} = \frac{1}{p+q+2}\int_C x^{p+1}y^q dy - x^p y^{q+1} dx \tag{3.48}$$

$$m_{00} = \frac{1}{2}\int_C x dy - y dx = \frac{1}{2}\int_C m_c x - [y_c + m_c(x - x_c)]dx \tag{3.49}$$

en la figura 3.6 adjunta se ve que el contorno de un objeto definido por el código de cadena permite expresar la integral curvilínea de (3.48) cómo un sumatorio.

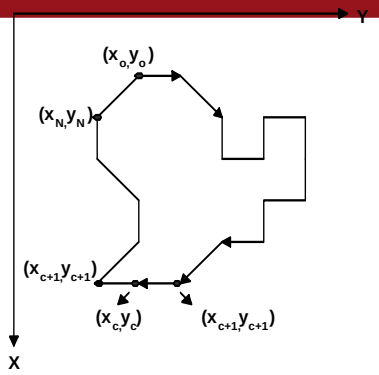



Figura 3.6 Curva del área

Por tanto, podemos escribir:

$$m_{00} = \frac{1}{2}\sum_{C=1}^{N}\int_{x_{c-1}}^{x_c} m_c x - [y_c + m_c(x - x_c)]dx \tag{3.50}$$

$$m_{00} = \frac{1}{2}\sum_{C=1}^{N}\int_{x_{c-1}}^{x_c} (m_c x_c - y_c)dx \tag{3.51}$$

obsérvese que el índice sumatorio se refiere al orden del enlace del código de cadena. Desarrollando la expresión (3.51):

$$m_{00} = \frac{1}{2}\sum_{C=1}^{N}(m_c x_c - y_c)(x_c - x_{c-1}) \tag{3.52}$$

$$m_{00} = \frac{1}{2}\sum_{C=1}^{N}(\frac{dy_c}{dx_c} x_c - y_c)dx_c \tag{3.53}$$

$$m_{00} = \frac{1}{2}\sum_{C=1}^{N}(x_c dy_c - y_c dx_c) \tag{3.54}$$

Es inmediato, dado el *código de cadena de un objeto*, obtener el sumatorio de la expresión (3.54) ya que se conoce las coordenadas de los pixeles y la siguiente tabla:

| **CC** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
|---|---|---|---|---|---|---|---|---|
| **$dx_c$** | 0 | -1 | -1 | -1 | 0 | +1 | +1 | +1 |
| **$dy_c$** | +1 | +1 | 0 | -1 | -1 | -1 | 0 | +1 |

Haciendo $Ac = x_c dy_c - y_c dx_c$, el momento de orden cero es:

$$m_{00} = \frac{1}{2}\sum_{C=1}^{N} A_c \tag{3.55}$$

Análogamente se puede obtener los momentos de mayor orden. La demostración de los siguientes resultados se muestra en el apéndice C.

$$m_{00} = \frac{1}{2}\sum_{c=1}^{N} Ac \tag{3.56}$$

$$m_{10} = \frac{1}{3}\sum_{c=1}^{N} A_c\left[x_c - \frac{dx_c}{2}\right] \tag{3.57}$$

$$m_{01} = \frac{1}{3}\sum_{c=1}^{N} A_c\left[y_c - \frac{dy_c}{2}\right] \tag{3.58}$$

$$m_{20} = \frac{1}{4}\sum_{c=1}^{N} A_c\left[x_c^2 - x_c dx_c + \frac{1}{3}dx_c^2)\right] \tag{3.59}$$

$$m_{02} = \frac{1}{4}\sum_{c=1}^{N} A_c\left[y_c^2 - y_c dy_c + \frac{1}{3}dy_c^2)\right] \tag{3.60}$$

$$m_{11} = \frac{1}{4}\sum_{c=1}^{N} A_c\left[x_c y_c - \frac{1}{2}(x_c dy_c + y_c dx_c) + \frac{1}{3}dx_c dy_c)\right] \tag{3.61}$$

$$m_{30} = \frac{1}{5}\sum_{c=1}^{N} A_c\left[(x_c - \frac{dx}{2})(x_c^2 - \frac{dx_c^2}{2} - x_c dx_c)\right] \tag{3.62}$$

$$m_{03} = \frac{1}{5}\sum_{c=1}^{N} A_c\left[(y_c - \frac{dy}{2})(y_c^2 - \frac{dy_c^2}{2} - y_c dy_c)\right] \tag{3.63}$$

# Capítulo 4

# Reconocimiento Automático de Patrones

## 4.1 Introducción

Una definición formal de Reconocimiento de Patrones es la siguiente: "es la categorización de datos de entrada en clases identificadas, por medio de la extracción de características significativas o atributos de los datos extraídos de un medio ambiente que contiene detalles irrelevantes". Matemáticamente hablando, la clasificación consiste en la partición del espacio n-dimensional definido por las características de un objeto, en varias regiones, donde cada región corresponde a una clase.

Un sistema de reconocimiento de patrones tiene uno de los siguientes objetivos:

a. Identificar el patrón como miembro de una clase ya definida (clasificación supervisada).
b. Asignar el patrón a una clase todavía no definida (clasificación no supervisada, agrupamiento o *clustering*).

El reconocimiento automático de formas gira entorno a varios conceptos claves que se definen algunos a continuación:

*a. Patrón:* un patrón es una descripción cuantitativa o estructural de un objeto o alguna entidad de interés**.**

*b. Clase*: una clase de patrones es un conjunto de patrones que comparten algunas propiedades.

*c. Vector de características:* la composición de varias características en un vector se denomina vector de características, esta contiene la medida de las características de un patrón; puede estar formado de números binarios o valores reales. Un vector característica define puntos en un espacio n-dimensional.

## 4.2 Enfoques de un Sistema de Reconocimiento

En la literatura pueden encontrarse diferentes enfoques para este problema, motivadas por la diversidad de tareas de reconocimiento que pueden abordarse. Sin embargo podemos identificar en general, 4 tipos de metodologías para el reconocimiento de patrones:

### 4.2.1 Heurísticas

Esta metodología es la que hace uso de la experiencia y la intuición humana. Por lo general, los sistemas que son desarrollados bajo estos métodos, están hechos a la medida del problema que se desea resolver.

### 4.2.2 Matemáticas

Este tipo de metodología hace uso de las propiedades comunes de los patrones y se basan en las reglas de clasificación formuladas en un marco matemático. Además, este enfoque se divide a su vez en otras dos categorías: las determinísticas y las estadísticas.

**a. Determinísticas**

No requieren de propiedades estadísticas, pero son limitadas. Entre los métodos determinísticos podemos encontrar, por ejemplo, la clasificación por distancia Euclidea.

**b. Estadísticas**

La aproximación más simple (y no por ello la menos eficiente) consiste en representar cada *patrón* mediante un *vector de números*, y cada *clase* por uno o varios *patrones prototipo*. Dado que existe variabilidad en las medidas registradas, cada componente del vector es una *variable aleatoria* y cada uno de sus valores es una realización de esa variable aleatoria.

Con esta aproximación un patrón no es más que un punto en el *espacio de representación de los patrones*, que es un espacio de dimensionalidad determinada por el número de variables consideradas. Esta aproximación concluye que es razonable que los patrones pertenecientes a una misma clase estén cercanos en el espacio de representación mientras que aquellos que pertenezcan a clases diferentes deberían estar en diferentes regiones del espacio

de representación. Dentro de los métodos estadísticos podemos encontrar a los clasificadores basados en ***las Regla de Clasificación de Bayes***.

### 4.2.3 Lingüísticas (sintácticas)

Las técnicas de este tipo reducen un objeto (habitualmente ya binarizado) a un conjunto de elementos estructurales o ***'primitivas'***. Si a esto unimos una sintaxis para relacionar estos elementos de forma espacial, obtenemos lo que se llama una representación sintáctica. Se trata de descomponer objetos complejos en términos de conjuntos de primitivas simples (como son arcos, ángulos, rectas, etc.) con reglas para describir la relación espacial entre ellas.

Hay que decir que es importantísimo el diseño de la sintaxis y de sus primitivas y se requiere habitualmente la especificación de un complejo conjunto de reglas que den una correspondencia unívoca entre la estructura y su representación. Además suelen surgir problemas de dependencia de la orientación de la estructura, tamaño, varias representaciones para una misma estructura, etc.

En general, a menor número de primitivas, más difícil es la representación y más largas son las cadenas de representación. Existen numerosos autores (Freeman, Badie-Shimura, Pavlidis) que idearon diferentes sintaxis según los objetos que querían reconocer.

### 4.2.4 Redes Neuronales Artificiales

Las Redes Neuronales tienen muchas similitudes con el reconocimiento estadístico de patrones (REP) concerniente en la representación de datos y los principios de clasificación. La implementación práctica es sin embargo muy diferente. El modo de análisis implica la configuración de una red de neuronas artificiales y el entrenamiento de la red para determinar como las neuronas individuales pueden afectar uno a la otra. El modo de reconocimiento implica el envío de datos a través de la red y la evaluación a que clase se aproximará mas.

## 4.3 Reconocimiento Estadístico de Patrones (REP).

El REP es una disciplina relativamente madura hasta el punto de que existen ya en el mercado sistemas comerciales de reconocimiento de patrones que emplean esta técnica. En REP, un patrón se representa por un vector numérico de dimensión *n*; de esta forma, un patrón es un punto en un espacio de características. *n*-dimensional.

Un REP funciona en dos modos diferentes: entrenamiento y reconocimiento. En *modo de entrenamiento*, se diseña el extractor de características para representar los patrones de entrada y se entrena al clasificador con un conjunto de datos de entrenamiento de forma que el número de patrones mal identificados se minimice. En el *modo de reconocimiento*, el clasificador ya entrenado toma como entrada el vector de características de un patrón desconocido y lo asigna a una de las clases o categorías.

El proceso de toma de decisiones en un REP se puede resumir como sigue. Dado un patrón representado por un vector de características

$$X = (x_1, x_2, ...x_n)^T \tag{4.1}$$

asignarlo a una de las **c** clases o categorías, $C_1$ , $C_2$, ..$C_c$. Decimos que la densidad o probabilidad condicional p(x | Ci ) es la probabilidad de que la variable aleatoria Ci, sea precisamente x. Dependiendo del tipo de información disponible sobre las densidades condicionales de las clases, se pueden diseñar varias estrategias de clasificación. Si todas las densidades condicionales p(x | Ci ), i=1,2,...c se conocen, la regla de decisión es la de Bayes que establece los límites entre las diferentes clases. Sin embargo, en la práctica las densidades condicionales no se conocen y deben ser estimadas (aprendidas) partiendo de los patrones de entrada. Si se conoce la forma funcional de estas densidades pero no sus parámetros, el problema se llama de toma de decisión ***paramétrico***. En caso contrario, estamos ante un problema de toma de decisión ***no paramétrico***.

### 4.3.1 Representación de los patrones

La entrada a un sistema de reconocimiento estadístico de patrones es un vector numérico que contiene los valores muestreados y cuantizados (o binarizados) de una serie de señales naturales. De una manera más formal, suponiendo patrones *n*-dimensionales, un **patrón** X es una variable aleatoria *n*-dimensional compuesta por *n* componentes $x_1$, $x_2$,..., $x_n$ tales que:

$$X = \begin{bmatrix} x_1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \tag{4.2}$$

Con esta aproximación un patrón no es más que un punto en el *espacio de representación de los patrones* que es un espacio de dimensionalidad determinada

por el número de variables consideradas. Esta aproximación concluye que es razonable que los patrones pertenecientes a una misma clase estén cercanos en el espacio de representación mientras que aquellos que pertenezcan a clases diferentes deberían estar en diferentes regiones del espacio de representación.

### 4.3.2 Similaridad entre Patrones.

La tarea fundamental de un sistema de reconocimiento de patrones (clasificador) es la de asignar a cada patrón de entrada una etiqueta. Dos patrones diferentes deberían asignarse a una misma clase si son *similares* y a clases diferentes si no lo son. La cuestión que se plantea ahora es la definición de una medida de similaridad entre patrones.

Supongamos un sistema de adquisición perfecto (sin ruido), podemos asegurar que:

a. La adquisición repetida del mismo patrón debería proporcionar la misma representación en el espacio de patrones. Por ejemplo, una cámara debería proporcionar siempre la misma imagen de la misma escena si las condiciones externas no cambiaran.
b. Dos patrones diferentes deberían proporcionar dos representaciones diferentes.
c. Una ligera distorsión aplicada sobre un patrón debería proporcionar una pequeña distorsión de su representación.

En definitiva, se supone que el proceso de adquisición es biunívoco y contínuo. Esta consideraciones sugieren que si las representaciones de dos patrones están muy cercanas en el espacio de representación, entonces los patrones deben tener un alto grado de similaridad. No obstante, no puede afirmarse tajantemente que a mayor distancia mayor disimilaridad ya que la medida (absoluta) de distancia depende de la escala en la que se cuantifiquen las variables asociadas al patrón.

### 4.3.3 Variabilidad entre patrones.

La suposición de un sistema de adquisición perfecto no deja de ser eso, una suposición. Los sistemas de adquisición introducen, indefectiblemente cierta distorsión o ruido, lo que produce una *variabilidad* en la representación de los patrones. Aunque es posible controlar eficientemente en muchos casos esta distorsión mediante el calibrado de los sistemas de adquisición aparece otra fuente de variabilidad por la propia naturaleza de los patrones.

Con mucha frecuencia, patrones de una misma clase difieren, incluso significativamente. Un ejemplo sencillo es el de los sistemas de reconocimiento de caracteres OCR que pueden interpretar diversos tipos de letra, incluso caracteres escritos (bajo fuertes restricciones). Bien en el caso más simple, en el que se trata un solo tipo de letra, la variabilidad de los patrones se debe a factores tales como el granulado, color y calidad del papel o el tipo de tinta empleado.

La variabilidad *intrínseca* de los patrones hace que las representaciones tengan la forma de *nubes de puntos* en lugar de puntos individuales (figura 4.1). Esta última representación sería la de darse el caso de que no existiera variabilidad entre los patrones. Estas nubes de puntos sugieren que patrones de una misma clase (similares) se representan cercanos (relativamente) en el espacio de representación mientras que patrones de diferentes clases (diferentes) se representan lejanos (relativamente) en ese espacio.

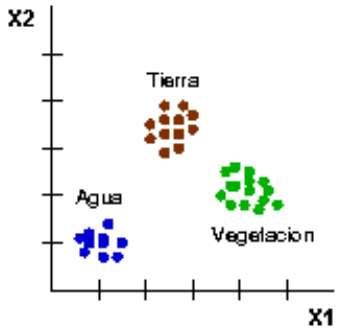



Figure 4.1 Variabilidad entre patrones.

Como vemos los patrones de una misma clase presentan cierta variabilidad natural. No obstante, deben estar (relativamente) cercanos en el espacio de representación y lejanos (relativamente) respecto a los patrones de otras clases.

Otra situación posible, y mucho más crítica, es la presencia de agrupamientos *solapados*. En la figura 4.2 mostramos la representación de un conjunto de patrones pertenecientes a dos clases. Los patrones son bidimensionales y las variables consideradas son medidas de peso y altura de un conjunto de personas. Las clases informacionales son *mujer* y *hombre.* En este caso no es fácil la discriminación de estas clases con estos patrones.

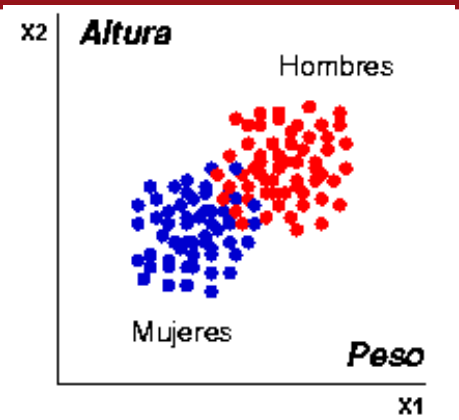



Figure 4.2 Agrupamientos solapados.

## 4.4 Etapas de diseño de un Reconocedor de Formas

a. La primera etapa consiste en el ***establecimiento de las clases***, en lo que podría denominarse como definición del ***universo de trabajo*** del sistema. En la mayoría de casos esta etapa es directa y trivial, el diseñador conoce perfectamente las clases de los objetos (piezas) que han de ser reconocidos.

   Si suponemos que todos los patrones a reconocer son elementos potenciales de N clases distintas denotadas, i = 1, 2,..., N, llamaremos:

   $$\Omega = \{\omega_1, \omega_2 \ldots, \omega_N\} \qquad (4.3)$$

   al conjunto de las ***clases***. Conviene tener en cuenta que una clase es la denominación que se da a una clase conocida y con significado. Por ejemplo, un conjunto de clases sensato para el problema de clasificaciones de zonas en una área terrena sería:

   $$\Omega = \{\text{agua, vegetación, suelo}\} \qquad (4.4)$$

   Resulta conveniente ampliar el conjunto Ω incorporando una nueva clase, llamada la ***clase de rechazo***. Así, se define la clase de rechazo $\omega_o$ como una clase que se asigna a todos los patrones para los que no se tiene una certeza aceptable de ser clasificados correctamente en alguna de las clases de Ω. Se dice que:

   $$\Omega^* = \{ \omega_1, \omega_2, \ldots, \omega_N, \omega_o \} \qquad (4.5)$$

   es el ***conjunto extendido de clases informacionales.***

   No obstante puede ocurrir que las clases puedan ser desconocidas a priori, en donde no esta clasificado el universo de clases. En tales situaciones se recurre a técnicas denominadas de agrupación (*clustering*) o reconocimiento no supervisado. Estas técnicas difieren de las expuestas con anterioridad, ya que no

existe una supervisión o reconocimiento externo que guíe el diseño de las funciones de discriminación.

b. Una vez definida las clases, la siguiente operación consiste en la ***elección del vector de características***. Esta etapa es crítica, y la bondad del sistema final estará completamente determinada por los rasgos (características) escogidos.

   El vector característico, sin duda, constituye el elemento clave en un sistema de reconocimiento automático de formas. La elección de rasgos o características es muy dependiente de la aplicación concreta que se tenga, se trata mas de un arte que de una ciencia y es la intuición y la experiencia quienes pueden guiar en su elección. En cualquier caso las características elegidas deben posibilitar que las clases sean disjuntas o separables.

c. Una vez seleccionado el vector característico del sistema, la etapa final es ya puramente mecánica y se centra en el cálculo de las correspondientes ***funciones de decisión o funciones discriminantes***. Dentro de las diferentes técnicas veremos solo algunas.

   Una vez establecido el conjunto de clases se procede a la construcción del clasificador. Como puede intuirse, la construcción del clasificador no es una tarea trivial ni directa e involucra una serie de etapas:

   - La elección del modelo.
   - Aprendizaje (entrenamiento del clasificador).
   - Verificación de los resultados.

Es muy importante señalar que estas etapas no deben verse de forma secuencial. Puede ocurrir que en un momento dado hay que volver atrás para replantearse alguno de los pasos dados, incluso el conjunto de clases informacionales. En la figura 4.3 observamos el diagrama de flujo para el diseño de un reconocedor automático.

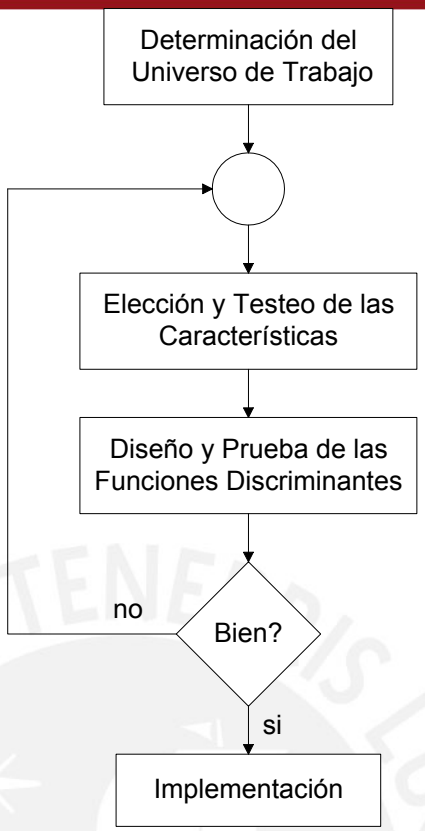



Figura 4.3 Diagrama de las etapas en el diseño de un Sistema de Reconocimiento Automático

## 4.5 Ejemplo de un Sistema de Visión Artificial

Supongamos que se pretende discriminar 3 tipos diferentes de frutas: melones, naranjas y fresas que pueden presentarse sobre una banda transportadora.

Entonces por lo visto hasta el momento, el proceso de diseño del correspondiente sistema consistiría en las siguientes etapas:

a. Establecimiento del universo de trabajo, es decir, las clases que se van a manejar $\alpha_1$, $\alpha_2$, $\alpha_N$,. En el ejemplo es inmediato que:

$$UT = \{melon, naranja, fresa\} \quad (4.6)$$

b. Elección y prueba de características. Aunque esta etapa suele ser delicada y compleja en la mayoría de aplicaciones reales, en el ejemplo podemos definir al menos dos de ellas. En primer lugar el perímetro de los diferentes patrones o clases son muy dispares, por lo que es un buen rasgo discriminante. Otro factor discriminante seria los niveles de intensidad de color de los tres tipos de frutas. Entonces escalando las características entre 0 y 1 se obtendría la distribución de las clases en el espacio abstracto (plano) de las dos anteriores características que se presentan en la figura 4.4:

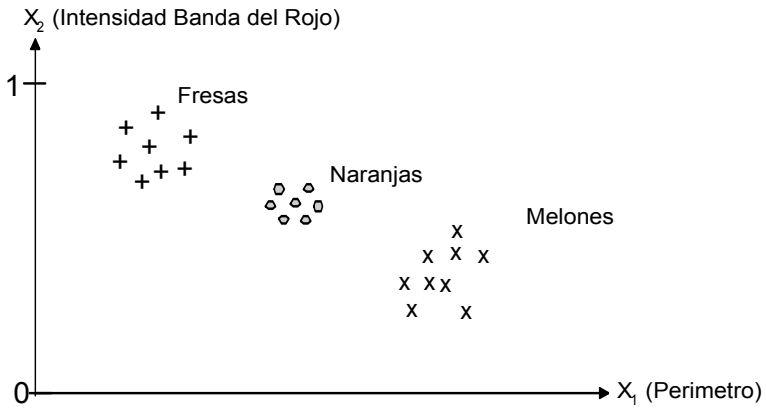



Figura 4.4 Distribución en el plano de las 3 clases del ejemplo

c. Cálculo de las funciones discriminantes. En este ejemplo es trivial trazar las rectas $W_1^T X$ y $W_2^T X$ que discriminen de forma inequívoca las 3 clases, ubicándolas en regiones separadas, como se verá mas adelante.

También se puede calcular las 3 funciones discriminantes basadas en la distancia Euclidea a los centroides de cada clase, como alternativa a la discriminación de regiones.

Resumiendo hasta este punto se han expuesto los conceptos básicos y fundamentales de la etapa de reconocimiento automático de formas: patrón, forma o clase, vector de rasgos o característico y función discriminante.

Las funciones discriminantes (ffdd) que se han manejado hasta ahora son lineales, es decir presentan la forma $W^T X$, esto porque las características son todas de separabilidad lineal, es decir pueden obtenerse ffdd lineales capaces de clasificar todas las clases sin ambigüedad.

Sin embargo en la distribución de clases que se presenta fuerte correlación, es decir no existe separabilidad lineal, es imposible tener una recta $W_1X_1 + W_2X_2 + W_3$ capaz de clasificar sin error las dos clases. En tales casos es necesario recurrir a ffdd no lineales, lo cual complica el diseño del clasificador automático. en estos casos es necesario recurrir a procedimientos as avanzados: discriminadores polinomiales, aprendizaje supervisado, redes neuronales, etc o bien escoger otro conjunto de características discriminadoras que conduzcan a un espacio abstracto de patrones con separabilidad lineal de las clases.

A continuación veremos los reconocedores en los cuales implica el aprendizaje.

## 4.6 Modelo de clasificadores

Se suele utilizar indistintamente los términos ***aprendizaje*** y ***entrenamiento*** para referirse al proceso de construcción del clasificador. El aprendizaje en los clasificadores puede realizarse de dos maneras muy diferentes: aprendizaje Supervisado y aprendizaje no Supervisado.

### 4.6.1 Aprendizaje Supervisado.

Un aprendizaje supervisado requiere disponer de un conjunto de patrones de los cuales se conoce su *clase cierta*. A este conjunto se le denomina **conjunto de entrenamiento**. Este tipo de entrenamiento se denomina **entrenamiento supervisado** y los clasificadores así obtenidos **clasificadores supervisados**.

En este caso se dispone de un conjunto de entrenamiento, los cuales han sido etiquetados previamente. Esta tarea la suele realizar un experto en el campo en el que se va a realizar el reconocimiento.

Dependiendo si se supone o no un completo conocimiento a priori de la estructura estadística de las clases el planteamiento a la hora de construir un clasificador es radicalmente diferente.

#### 4.6.1.1 Modelo Paramétrico

Suponemos que podemos modelar las clases mediante funciones de densidad de probabilidad conocidas: esta forma de abordar el problema se conoce como ***aproximación paramétrica***. Como veremos, el aprendizaje se reducirá a la estimación de los parámetros que determinan las funciones de densidad de probabilidad de las clases. La herramienta fundamental que se usa será la *estadística* y particularmente la *teoría de la decisión de Bayes*

Si se supone un completo conocimiento a priori de la estructura estadística de las clases, el aprendizaje se reduce a la estimación de los parámetros que determinan las funciones de densidad de probabilidad de las clases. Los clasificadores construidos bajo esta suposición se conocen como ***clasificadores paramétricos***.

En este planteamiento, los patrones de una clase dada son realizaciones de una distribución de probabilidad normal, cuyos parámetros son el vector medio y la matriz de covarianza. Cada punto del espacio de representación tiene asociado un valor de densidad de probabilidad de pertenencia a cada clase.

Se dice que un punto *X* está en la región de decisión asociada a una clase si el valor de probabilidad de pertenencia a esa clase asociado a *X* es mayor que para cualquiera otra clase.

#### 4.6.1.2 Modelo no Paramétrico

Si no se supone un determinado modelo estadístico, bien por desconocimiento o por la imposibilidad de asumir un modelo paramétrico adecuado, el problema resulta más complejo y se puede abordar desde diferentes perspectivas. Los clasificadores construidos sin esta suposición se conocen como ***clasificadores no paramétricos.***

Cuando no puede suponerse un modelo paramétrico para las funciones de densidad de probabilidad asociadas a las clases se utilizan modelos no paramétricos para estimar las funciones de densidad de probabilidad. En estos casos la única información disponible para la clasificación es el conjunto de prototipos. El método más simple entre los no paramétricos es el del *vecino más cercano*, que consiste en etiquetar un patrón con la etiqueta del prototipo más cercano.

### 4.6.2 Aprendizaje no Supervisado.

El aprendizaje no supervisado se realiza a partir de un conjunto de patrones del que no se conoce su clase cierta. Básicamente, se traduce en *encontrar agrupamientos*. El objetivo suele ser el de verificar la validez del conjunto de clases informacionales para una clasificación supervisada. Las técnicas utilizadas suelen denominarse **métodos de agrupamiento** o **clustering**.

## 4.7 Reconocedor Estadístico.

En situaciones en los cuales los vectores de algunas clases presentan una dispersión significativa con respecto a su media (figura 4.1) es conveniente abandonar la hipótesis determinista a favor del enfoque estadístico.

El teorema de Bayes puede expresarse de la siguiente manera:

$$p(\alpha_i / X) = \frac{p(X/\alpha_i).p(\alpha_i)}{p(X)} \qquad (4.7)$$

donde:

$p(\alpha_i/X)$ : probabilidad de que un vector características X pertenezca a la clase $\alpha_i$

$p(X/\alpha_i)$: probabilidad de que $\alpha_i$ , el valor de la variable aleatoria sea precisamente X. Dicho en otras palabras es la fdp de la clase $\alpha_i$ considerada como una variable aleatoria.

$p(\alpha_i)$ : probabilidad a priori de que se presente un elemento de la clase $\alpha_i$

$p(X)$ : probabilidad a priori de que se presente un objeto a clasificar con un vector de características igual a X (considerado como un vector numérico concreto)

Este ultimo puede despreciarse, ya que presenta el mismo valor para un conjunto de N clases $\alpha_1$ , $\alpha_2$ ,… $\alpha_N$ compitiendo por el vector X a clasificar.

El primer término de la ecuación () aporta la solución al problema de la clasificación. Dado un vector X a que clase $\alpha_1$ , $\alpha_2$ ,… $\alpha_N$ pertenece? obviamente la clasificación será la siguiente:

$$X \in \alpha_j \quad si \quad p(\alpha_j/X) > p(\alpha_i/X) \qquad \forall i \neq j,\ i=1,2,..N \qquad (4.8)$$

Si ahora nos basamos en el segundo miembro de la expresión del teorema de Bayes tendremos una forma alternativa de clasificar un vector X.

$$X \in \alpha_j \quad si \quad p(X/\alpha_j).p(\alpha_j) > p(X/\alpha_i).p(\alpha_i) \quad \forall i \neq j,\ i=1,2,..N \qquad (4.9)$$

Adoptamos el calificativo a posteriori al clasificador estadístico basado en el primer miembro del teorema de Bayes y el calificativo a priori basado en el segundo miembro. La estimación de las probabilidades del clasificador estadístico a posteriori solo se puede realizar mediante un proceso de aprendizaje (aprendizaje no paramétrico)

En cuanto al otro procedimiento el principal problema para el diseño del clasificador a priori es la estimación estadística de las funciones de densidad de probabilidad de las clases $p(X/\alpha_1)$, $p(X/\alpha_2)$, …$p(X/\alpha_N)$,a partir de un conjunto de muestras de las clases $\alpha_1$, $\alpha_2$,… $\alpha_N$ consideradas como variables aleatorias.

### 4.7.1 Clasificador Bayesiano

Este clasificador, al que vamos a referirnos indistintamente también como clasificador bayesiano se basa en manejar el segundo miembro del teorema de Bayes:

$$p(X/\alpha_i).p(\alpha_i) \qquad i=1,2,\ldots N \qquad (4.10)$$

que son las funciones de densidad de probabilidad de las diferentes clases $\alpha_1,\alpha_2,\ldots\ \alpha_N$ Restringiremos nuestro estudio al caso de hipótesis de distribuciones normales o gaussianas, que es la que se dan en la mayoría de los casos prácticos.

Si se tienen dos clases y son equiprobables, es decir $p(\alpha_1) = p(\alpha_2)$, además el vector características es bidimensional, entonces la distribución gausiana tiene la forma:

$$p(X/\alpha_i) = \frac{1}{\sqrt{2\pi\sigma_{i1}\sigma_{i2}}} \exp\left[-0.5\left[\frac{(X-m_{i1})^2}{\sigma_{i1}^2} + \frac{(X-m_{i2})^2}{\sigma_{i2}^2}\right]\right] \qquad (4.11)$$

donde $m_{i1}$, $m_{i2}$ son las medias de las características $X_1$ y $X_2$ para la clase $\alpha_i$. Obviamente $\sigma_{i1}$ y $\sigma_{i2}$ son sus desviaciones típicas. En este caso se supone que ambas características están incorreladas.

Para el caso general de un vector de características n-dimensional, la fdp sigue la siguiente ley:

$$p(X/\alpha_i) = \frac{1}{(2\pi)^{n/2}|Ci|^{1/2}} \exp\left[-0.5(X-m_i)^T C_i^{-1}(X-m_i)\right] \qquad (4.12)$$

en donde $m_i$ y $C_i$ son respectivamente el vector media o esperanza matemática y la matriz de covarianza de la clase $\alpha_i$, considerada esta como una variable aleatoria

### 4.7.2 Ejemplo de un Clasificador Bayesiano

Se plantea el problema de reconocimiento entre la letra mayúscula B y el numero 8. Al estar escrito a mano y posiblemente por diferentes individuos, hace que las características discriminantes presenten aleatoriedad.

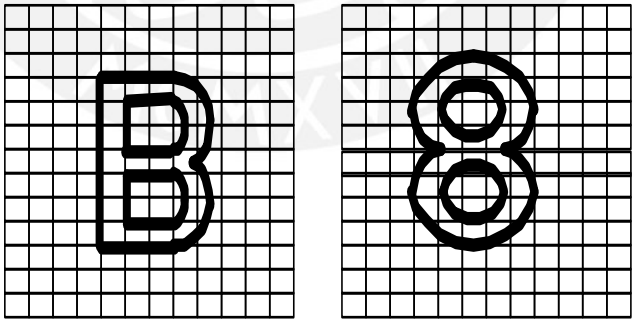



Figura 4.5 Caracteres a Reconocer: B y 8.

Analicemos la figura para en donde se presentan los mapas de bits de los dos caracteres a distinguir. Se puede apreciar que los trazos en sentido vertical de la parte izquierda de ambos caracteres difieren claramente. Así en el trazo del 8 es muy curvo, mientras que el trazo e la B es muy lineal. Esto nos permite definir una característica discriminante X1 llamado grado de linealidad:

$$X_1 = \left[ \frac{\text{numero de pixels del trazo izquierdo}}{\text{numero de filas entre pixeles extremos}} \right]^{-1}$$

es decir se trata de dividir el numero de pixeles del contorno del trazo vertical izquierdo por el numero de filas entre el primero y el ultimo pixel de dicho contorno

Otra característica es el tamaño relativo de las parte superior e inferior de los caracteres:

$$X_1 = \left[ \frac{\text{area parte superior}}{\text{area parte inferior}} \right]$$

Una posible distribución de las muestras en el espacio abstracto de las clases se observa en la figura, donde se comprueba dos hechos, la naturaleza aleatoria de ambas clases y el evidente solape entre ellas.

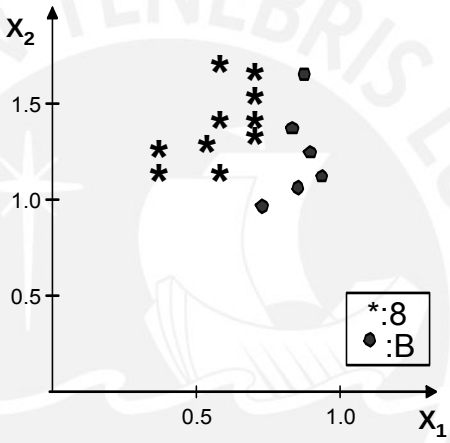



Figura 4.6 Representacion de muestras de ambos caracteres

Como vemos no existe separabilidad lineal entre las dos clases, veremos como el enfoque estadístico permite obtener una mejora en la taza de clasificación respecto a los clasificadores determinísticos lineales.

La regla de decisión del clasificador Bayesiano es:

$$p(X/B).p(B) > p(X/8)p(8) \Rightarrow \quad X \in B \tag{4.13}$$

Las fdp serán:

$$p(X/B) = \frac{1}{2\pi\sigma_1\sigma_2} \exp\left[ -0.5 \left[ \frac{(X_1 - m_{B1})^2}{\sigma_1^2} + \frac{(X_2 - m_{B2})^2}{\sigma_2^2} \right] \right] \tag{4.14}$$

$$p(X/8) = \frac{1}{2\pi\sigma_1\sigma_2} \exp\left[ -0.5 \left[ \frac{(X_1 - m_{81})^2}{\sigma_1^2} + \frac{(X_2 - m_{82})^2}{\sigma_2^2} \right] \right] \tag{4.15}$$

No obstante para llegar a una interpretación geométrica del clasificador Bayesiano expresaremos p(X/B) y p(X/8) en su forma general:

$$p(X/B) = \frac{1}{2\pi\sigma_1\sigma_2}\exp\left[-0.5(X-m_B)^T C_B^{-1}(X-m_B)\right] \quad (4.16)$$

$$p(X/8) = \frac{1}{2\pi\sigma_1\sigma_2}\exp\left[-0.5(X-m_8)^T C_8^{-1}(X-m_8)\right] \quad (4.17)$$

en donde:

$$C_B = C_8 = C = \begin{pmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{pmatrix} \quad (4.18)$$

en virtud a la hipótesis p(B)=p(8), la clasificación se efectúa a partir de la función discriminante:

$$(X-m_B)^T C^{-1}(X-m_B) = (X-m_8)^T C^{-1}(X-m_8) \quad (4.19)$$

esta expresión equivale a:

$$X^T C^{-1}(m_B - m_8) = \frac{1}{2}(m_B + m_8)C^{-1}(m_B - m_8) \quad (4.20)$$

que es la ecuación de la recta que atraviesa el punto $\frac{1}{2}(m_B + m_8)$. Resolviendo se obtendría la función discriminante de la figura 4.7. En donde podemos apreciarse la sensible mejora lograda por el clasificador Bayesiano comparado con la hallada en forma determinística.

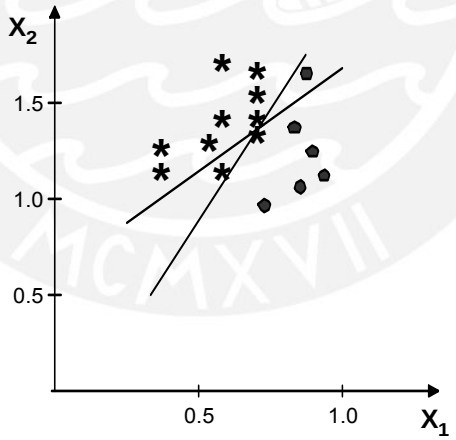



Figura 4.7 Nueva función discriminante cuando las desviaciones tipicas de las caracteristicas difieren

# Capítulo 5

# Red Neuronal como Reconocedor de Patrones

## 5.1 Introducción

Tradicionalmente el reconocimiento de patrones se ha abordado desde un punto de vista estadístico, dando lugar al llamado reconocimiento estadístico de patrones (REP), del cual ya se mencionó anteriormente. No obstante, existe una alternativa que se ha revelado como muy prometedora en algunos casos en que el REP no funciona satisfactoriamente; dicha alternativa son las Redes Neuronales Artificiales (RNA).

## 5.2 Redes Neuronales

La definición más general considera una RNA como una estructura formada por muchos procesadores simples llamados *nodos* o *neuronas*, conectados por medio de canales de comunicación o *conexiones*. Cada una de ellas tiene una cantidad de memoria local, operando solamente con sus datos locales y sobre las entradas que recibe a través de esas conexiones.

Las RNAs llevan asociadas algún tipo de *regla de aprendizaje* o *entrenamiento* particular por la cual esas conexiones son ajustadas acorde a los ejemplos proporcionados. En otras palabras, las RNAs aprenden a partir de ejemplos, y muestran alguna capacidad para *generalizar* más allá de esos datos mostrados.

De manera general las redes neuronales presentan tres principales características: *aprendizaje*, *generalización*, *adaptabilidad*. La característica de *aprendizaje* se refiere a que al igual que el cerebro humano, una red neuronal tiene la capacidad de almacenar conocimiento mediante un proceso de aprendizaje (entrenamiento). Este conocimiento es almacenado por los pesos de las conexiones entre los neurones que conforman a la red neuronal. Por otra parte, la capacidad de *generalización* significa

que se pueden obtener salidas razonables cuando se usan entradas diferentes a las utilizadas durante el proceso de entrenamiento. Por último, la característica de *adaptabilidad* significa que una red neuronal puede ser reentrenada para funcionar adecuadamente ante cambios en su medio ambiente.

En la mayoría de las RNAs propuestas, cada neurona procesa la información según el modelo propuesto por McCulloch-Pitts.

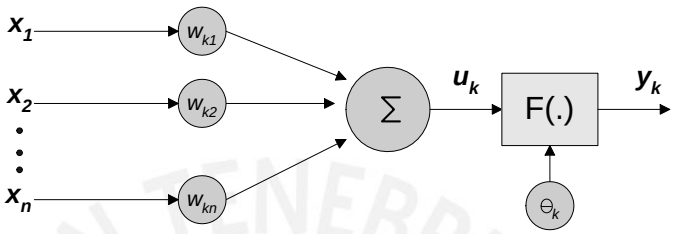



Figura 5.1 Neurona Artificial (McCulloch-Pitts)

$$y_k = F(X) = F(x_1 w_1 + x_1 w_1 + \ldots x_n w_n) \tag{5.1}$$

Cada neurona obtiene su salida como la suma ponderada de cada una de sus entradas ($x_1$, $x_2$.. $x_n$) previa multiplicación por su correspondiente peso ($w_1$,$w_2$,.. $w_n$). Desde el punto de vista matemático, la salida así calculada puede alcanzar cualquier magnitud; sin embargo, es sabido que en la naturaleza los valores de los potenciales electroquímicos están acotados. Para dar cuenta de este hecho, la salida obtenida es filtrada por una función de activación o función de transferencia F(x), que bien podría tratarse de una función salto [0,1] desplazada del origen un cierto umbral. Pero no es la función salto la más comúnmente empleada sino otra con un perfil parecido: *la sigmoide*. La razón es de conveniencia numérica, ya que es una función *n*-diferenciable.

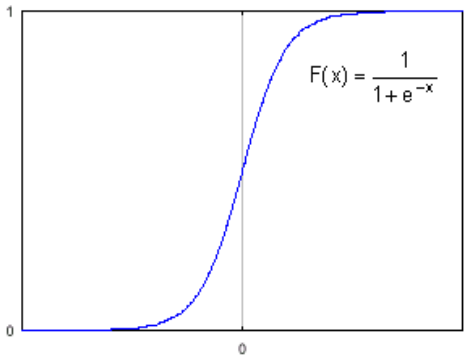



Figura 5.2Función Sigmoide

Como es sabido, numéricamente presenta la ventaja de que su derivada se puede expresar mediante la propia función:

$$F(x) = \frac{1}{1+e^{-x}} \Rightarrow F'(x) = F(x)(1-F(x)) \quad (5.2)$$

Se pueden utilizar otras funciones diferentes a la sigmoide, con la condición de que sean continuas y diferenciables en cualquier punto, a fin de que pueda obtenerse su gradiente.

### 5.2.1 Arquitectura

Básicamente podemos decir que entre las topologías o arquitectura más conocidas sobre redes neuronales tenemos:

a. Perceptrones multicapa o de una sola capa
b. Redes recurrentes y de Hopfield
c. Redes de Kohonen

Una de las arquitecturas mas utilizadas para el reconocimiento de patrones es la red multicapas o red de perceptrones multicapas (MLP).

#### a. Red Multicapas

Básicamente, este tipo de arquitectura de red consta de tres componentes principales: una capa de entrada, una o más capas ocultas, y una capa de salida. Este tipo de redes comúnmente son entrenadas utilizando el algoritmo de retropropagación de error el cual consta de dos fases. La primera, es una fase hacia delante, en la cual la información colocada en los neurones de la capa de entrada (patrón) es propagada hacia delante a través de las capas ocultas y hasta la capa de salida, la cual genera en los neurones que la conforman, la respuesta a la información dada como entrada. Durante esta fase, todos los pesos de las conexiones entre neurones son utilizados. La segunda fase es conocida como fase hacia atrás, la cual consiste en modificar los valores de los pesos de acuerdo al error generado por los neurones de la capa de salida.

Retropropagación es una técnica de minimización del error que aplica el concepto de gradiente descendente. Su objetivo es minimizar la función del error promedio al cuadrado que hay entre la salida real y la salida deseada de la red.

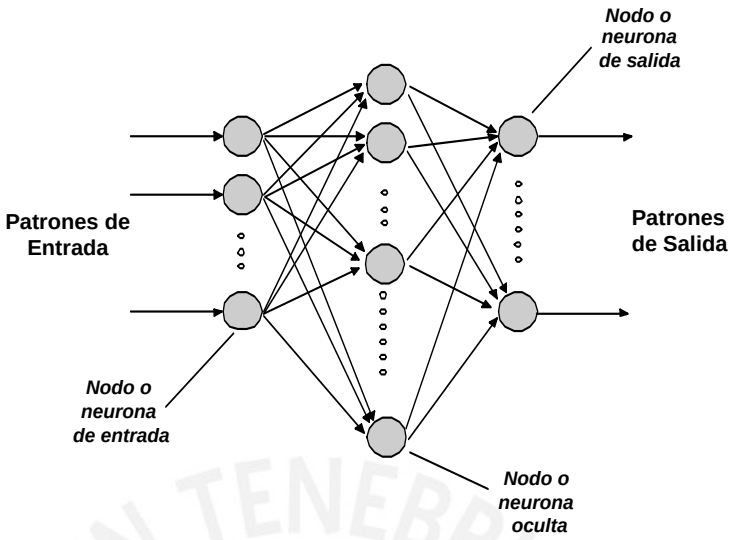



Figura 5.3 Perceptron Multicapa

### 5.2.2 Aprendizaje

En cada tipo de red existe al menos un algoritmo de aprendizaje, el cual consiste en un método para encontrar un valor adecuado de los pesos. En términos generales, se basan en definir una función objetivo implícita o explícitamente, que represente de forma global el estado de la red. A partir de ella, los pesos asignados inicialmente van evolucionando a unos valores que lleven a dicha función a un mínimo (estado estable de la red). El aprendizaje, por tanto, es algo característico del tipo de red.

#### 5.2.2.1 Reglas básicas de aprendizaje.

También se denominan *algoritmos de aprendizaje* cuando se posee una formulación matemática particular según el modo de actualización de los pesos sinápticos de la red. Cada regla, por tanto, tendrá unas ventajas específicas respecto las otras, dependiendo entre otros factores, de la aplicación.

a. Aprendizaje por corrección del error.
b. Aprendizaje de Hebb o Hebbiano.
c. Aprendizaje de Boltzmann.
d. Aprendizaje competitivo.

*a. Aprendizaje por corrección del error:* En principio la señal proporcionada por una neurona de salida es diferente de la respuesta deseada para esa neurona por lo que es posible definir una *señal de error*. El propósito de este tipo de aprendizaje

consiste en minimizar una *función de error* o *coste* basada en ese error cometido. Un criterio muy usado es el *error cuadrático medio.* La corrección de los pesos con ese fin es proporcional a la señal de error y a la de entrada.

#### 5.2.2.2 Esquemas de aprendizaje.

Las RNAs pueden ser clasificadas atendiendo a su esquema conceptual en dos bloques: las RNAs *supervisadas* y las *no supervisadas.*

El *aprendizaje supervisado* maneja pares entrada–salida deseada. De esta manera, para una determinada entrada se compara la salida obtenida por la red con la deseada. El ajuste de los pesos está dirigido, por tanto, a minimizar en lo posible la diferencia entre dichas salidas. Un ejemplo muy típico de aprendizaje supervisado corresponde al de "retropropagación del error" para el perceptrón multicapa.

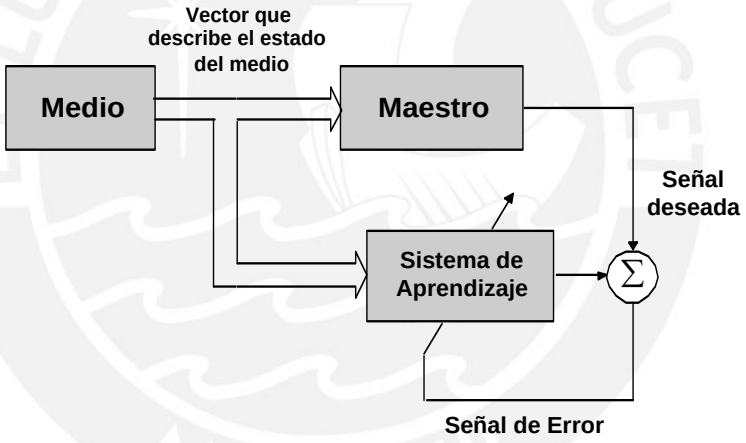



Figura 5.4 Diagrama de Bloques del Aprendizaje Supervisado

### 5.2.3 Generalización.

Diremos que una red neuronal es capaz de *generalizar* cuando existe una estrecha relación entre la salida deseada y la salida calculada por la red para muestras o ejemplos que nunca han sido usados para entrenarla.

Esto es, dispondremos de un conjunto de datos, ejemplos o patrones para ajustar los pesos de la red en clave a un determinado algoritmo de aprendizaje (conjunto de entrenamiento), así como de un conjunto de datos, ejemplos o patrones que la red nunca ha utilizado (conjunto de generalización). La generalización será la cualidad

que poseen las RNAs para extraer información de los datos presentados en su entrenamiento y extender ese conocimiento más allá de esas situaciones conocidas.

El proceso de aprendizaje (entrenamiento de la red neuronal) puede ser visto como un problema de ajuste de una curva. La red puede ser considerada simplemente como un mapeador no lineal de datos entrada-salida.

En el aprendizaje podemos cometer el error de entrenar en exceso la red de modo que ésta *memorice* y se comporte como una tabla sin ninguna capacidad para extraer información de los datos de test presentados. De esta manera, la generalización no siempre es posible a pesar de las aserciones de algunos autores.

## 5.3 Red Neuronal como Clasificador

En REP, el objetivo es asignar un patrón de entrada x, de dimensión n a una de las c clases o categorías, $C_1$, $C_2$,...$C_c$. Si las densidades *condicionales* $p(x/C_i)$ y las c densidades a priori, $p(Ci)$ se conocen, la estrategia óptima de clasificación es la regla de Bayes: asignar x a la clase $C_k$ si se cumple la condición:

$$p(\frac{C_k}{x}) > p(\frac{C_i}{x}) \qquad i = 1, 2,...c,\ i \neq k \qquad (5.3)$$

donde $p(C_k \mid x)$ se llama la probabilidad *a posteriori* y se calcula por la ley de Bayes. En REP una regla de decisión se puede formular a menudo en forma de una función discriminante $g_i(x)$, i=1...c. Se asigna x a la clase k si $g_k(x) > g_j(x)$ , $j \neq k$.

Algunas RNAs muy usadas, como el perceptrón multicapa o las funciones de base radial, en realidad calculan funciones discriminantes no lineales. En REP la forma de calcular las funciones discriminantes es bastante más complicada e implica establecer estimaciones de las densidades condicionales.

El perceptrón multicapa es un clasificador óptimo en el sentido de Bayes si las densidades condicionales de las clases son gaussianas con matrices de covarianza idénticas. Si estas premisas no se cumplen, el funcionamiento del perceptrón puede ser muy pobre y es necesario tomar algunas medidas adicionales para mejorarlo. En muchos problemas de REP no se conoce ni siquiera la forma de las densidades condicionales. En estos casos o bien se postula una forma, o se diseñan métodos no paramétricos como los clasificadores de los k vecinos.

La capacidad de generalización de los clasificadores, clásicos o basados en RNA, depende básicamente de tres factores:

a. Número de patrones de entrenamiento.

b. Las densidades condicionales presentes en los datos.

c. La complejidad del clasificador.

Las relaciones son más o menos conocidas si las distribuciones son gaussianas. En caso contrario, el problema de determinar las tasa de generalización de un clasificador es intratable y el método de prueba y error es el único disponible. Intuitivamente, parece obvio que cuantas más muestras de entrenamiento se utilicen mejor será el clasificador; pero no siempre se puede disponer de muchas muestras y por este motivo se están desarrollando técnicas que mejoren las tasas de generalización de una red neuronal, que son similares a las empleadas para seleccionar el mejor clasificador de un conjunto dado en REP.

Las redes neuronales en su calidad de aproximadores universales pueden construir arbitrariamente bien cualquier mapa del espacio de medidas (características) al espacio de las decisiones (clases). En particular, bajo el criterio de optimización de mínimos cuadrados, las redes neuronales pueden aproximare al clasificador bayesiano, y por tanto construir clasificadores óptimos.

## 5.4 Ventajas de los Redes Neuronales frente a los REP.

Entre las razones de la amplia utilización de las redes neuronales, se pueden destacar las siguientes:

a. Teóricamente pueden determinar cualquier función, por lo que son adecuadas en aplicaciones que no son fácilmente descritas analíticamente.

b. Excepto los patrones de entrada, no es necesario suministrar información adicional.

c. Se pueden aplicar a cualquier tipo de patrones y a cualquier tipo de datos.

d. Se obtienen buenos resultados con datos ruidosos, como los encontrados frecuentemente en meteorología.

e. No se hacen hipótesis acerca de la distribución estadística de las variables de entrada, es decir los modelos neuronales no necesitan un conocimiento a priori de la distribución estadística de los datos y sus clases

a. Los modelos neuronales no necesitan un conocimiento a priori de los parámetros internos del clasificador.

f. Después de entrenadas son extremadamente rápidas y fácilmente implementables en arquitecturas paralelas.

A menudo, las propiedades estadísticas de las clases de patrones de un problema son desconocidas, o no es posible realizar una estimación de las mismas. En la práctica, estos problemas de decisión se manejan mejor utilizando métodos que obtienen directamente las funciones de decisión requeridas mediante el entrenamiento. Así pues, no es necesario realizar suposiciones sobre las funciones de densidad de probabilidad subyacentes, o sobre cualquier otra información estadística relativa a las clases de patrones consideradas.

## 5.5 Pasos para la Implementación de una Red Neuronal como Clasificador

A continuación se entregan una serie de recomendaciones prácticas para construir un clasificador:

**a. Construir una Base de Datos.**

- Número de ejemplos suficientemente grande (al menos 100 por clase).
- Selección de variables: descartar aquellas irrelevantes reducir la dimensión del espacio de entradas a través de un análisis de componentes principales o mediante la selección de un subconjunto de características.

**b. Separar los datos.**

- Dividir los datos en: conjunto de entrenamiento, validación y prueba, en una proporción 50%, 25% y 25% aproximadamente. Para base de datos pequeñas (<1000 casos) el desempeño puede variar con la elección del conjunto de entrenamiento.
- Para base de datos pequeñas se puede usar el método de validación cruzada: los datos se dividen en n porciones y se entrenan n veces, cada vez dejando fuera un conjunto diferente para prueba.

**c. Transformar los datos de entrada en valores apropiados para la red.**

- Datos numéricos: normalizar entre 0 y 1 para unidades sigmoidales y -1 y 1 para unidades tangente hiperbólica. Normalizar con la misma escala diversas entradas de modo de evitar sesgos hacia aquellas con magnitudes mas grandes.

**d. Seleccionar el tipo de red neuronal**

- Retropropagacion del error (la mas popular) u otros(LVQ,RBF).

**e. Seleccionar arquitectura de red neuronal**

- Utilizar una o mas capas ocultas.
- Numero de unidades ocultas. Empíricamente se sugiere comenzar con Nh=(Ni+No)/2; donde Nh es el numero de unidades en la capa oculta, Ni es el numero de entradas y No es el numero de salidas.
- Tamaño depende de la complejidad de la frontera de clasificación.
- Función de transferencia (logística, tangente hiperbólica)
- Regla de aprendizaje (on line, batch)
- Taza de aprendizaje(fija, adaptativa)
- Función de costos (error cuadrático medio)

**f. Entrenar**

- Medir el error cuadrático medio.
- Evaluar el desempeño en conjuntos de entrenamiento y validación. Un clasificador bien entrenado tiene el mismo error para entrenamiento y validación.
- Usar detención temprana o regularización.
- Alternativamente, utilizar algoritmos de crecimiento, poda o algoritmos genéticos.

**g. Analizar Resultados**

- Matriz de confusión (análisis de errores basado en las confusiones entre clases)
- Histograma de pesos.

# Capítulo 6

# Sistema Robótico ER-IX

## 6.1 Introducción

En este capitulo veremos algunas características del robot Scorbot ER-IX, el cual utilizaremos. Dicho manipulador consta de 5 grados de libertad, sin incluir la pinza, cuyos movimientos son controlados mediante un controlador, el cual es el encargado de instruir los movimientos del robot, monitorear esos movimientos y hacer ajustes automáticamente para corregir cualquier error.

## 6.2 Trayectoria de control

Para mejor performance, la trayectoria de control puede ser programada dentro del sistema de control. Se ofrece dos trayectorias: paraboloide y senoidal.

La trayectoria paraboloide causa que los motores aceleren lentamente hasta que se logre una máxima velocidad, luego desacelera a la misma razón

La trayectoria senoidal causa a los motores acelerar y desacelerar rápidamente al inicio y al final del movimiento, con una velocidad constante a lo largo de la ruta. Esto se puede apreciar en la figura 6.1

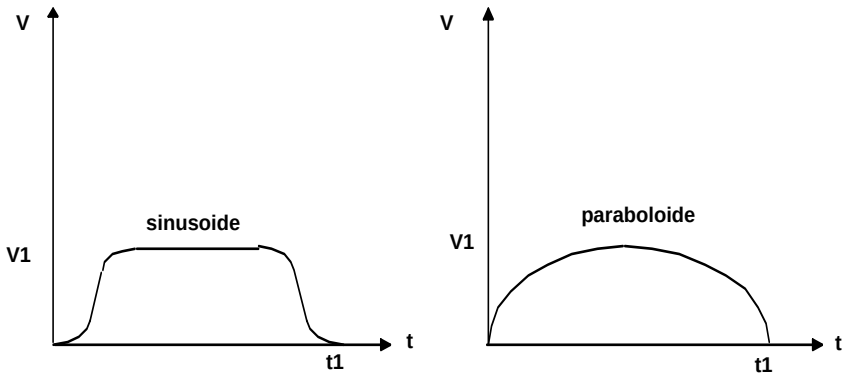



Figura 6.1Control de trayectorias

## 6.3 Programación ACL del Sistema Robótico.

ACL, Lenguaje de Control Avanzado, es un lenguaje de programación robótico multitarea avanzado desarrollado por Eshed Robotec. El programa ACL está almacenado sobre un conjunto de EPROMs dentro del controlador y puede ser accesado desde cualquier terminal estándar o computadora por medio de un canal de comunicación RS232.

ATS, Software de terminal Avanzado, es la interface usuario al controlador ACL y opera sobre cualquier computadora. El software ATS es un emulador de terminal el cual habilita accesar al ACL desde una computadora.

El siguiente diagrama muestra los componentes del sistema de control robótico.

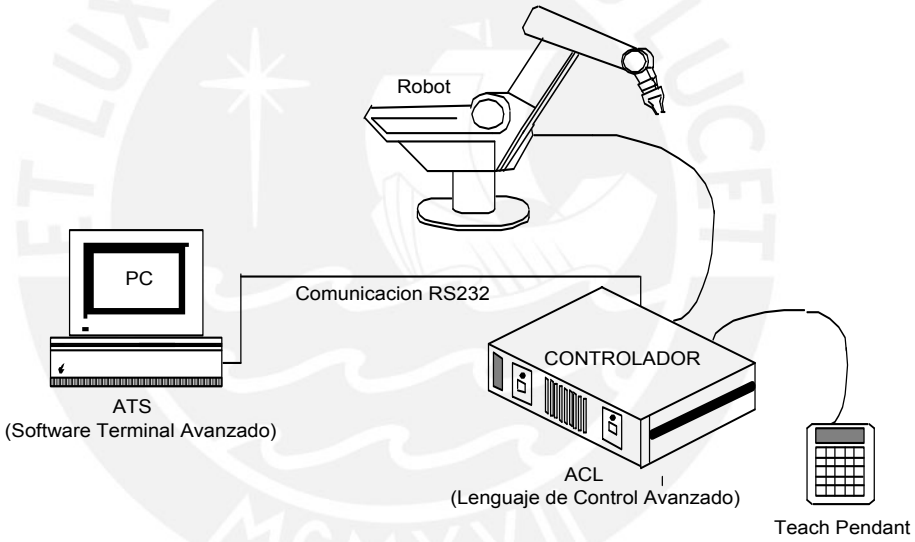



Figura 6.2 Diagrama del Sistema robótico

Características del ACL incluyen lo siguiente:

a. Control por el usuario en forma directa de los ejes del robot.
b. Programación del usuario del sistema robótico.
c. Control de datos de entrada/salida.
d. Ejecución del programa simultáneo y sincronizado (soporte multitarea).

ACL tiene comandos agrupados de acuerdo a categorías, de las cuales las mayormente usadas son listadas a continuación:

a. Comandos de control de ejes.
b. Comandos de control de I/O.

c. Comandos de control de programa.
d. Comandos de manipulación y definición de posiciones.
e. Comandos de manipulación y definición de variables.
f. Comandos de flujo de programa.
g. Comandos de comunicación RS232.

ACL tiene 2 tipos de comandos:

Los comandos ***Direct*** son ejecutados tan pronto como ellos son ingresados en el teclado del terminal/computadora.
Los comandos ***Edit*** o indirecto son ejecutados durante la ejecución de los programas y rutinas en el cual son usados.
Algunos comandos pueden ser usados en ambos modos ***Directo*** y ***Edit***.

## 6.4 Sistema de Coordenadas

El sistema robótico puede ser operado y programado en dos diferentes sistemas de coordenada: coordenadas Joint(encoder) y coordenadas Cartesianas (XYZ).
El sistema de coordenada cartesiana o XYZ es un sistema geométrico usado para especificar la posición del TCP del robot (Punto Central del Terminal) definiendo su distancia, en unidades lineales, desde el punto del origen (centro de la base del robot) a lo largo de los tres ejes lineales, como se ve en la figura 6.3. Para completar la definición de las posiciones, la elevación(pitch) y el giro(roll) de la pinza se especifican en unidades angulares.
Cuando se ejecuta el movimiento del robot en modo XYZ, todos o algunos de los ejes se mueven para mover el TCP a lo largo de un eje X, Y y/o Z.

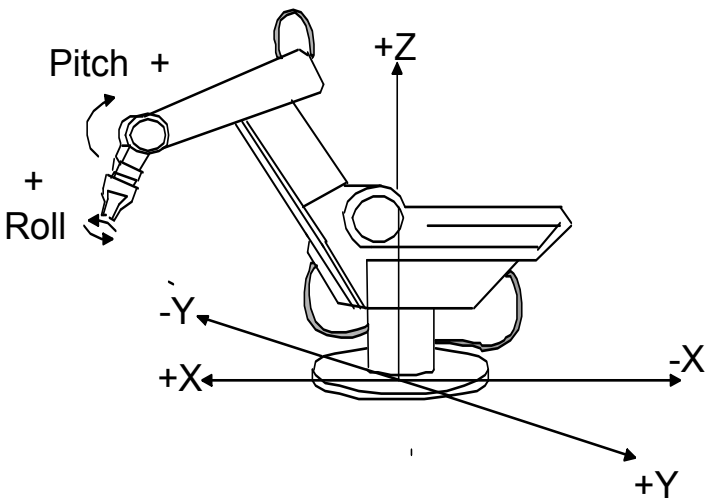


Figura 6.3 Ejes de coordenadas cartesianas del Robot

## 6.5 Sincronización entre las coordenadas del Robot-Frame.

La fase de sincronización está destinada a establecer la correspondencia entre el sistema visual y del robot; es decir, transforma las posiciones en el sistema de coordenadas de visión (cámara) a posiciones en el sistema de coordenadas del robot, permitiendo así manipular objetos que han sido identificados.

Consideremos el caso en que el cuadro de captura (frame) por la cámara es paralelo al plano z de las coordenadas del robot. Si esto es así, entonces para poder transformar las coordenadas en el plano imagen a las coordenadas del espacio de trabajo del robot, en forma general deberemos realizar transformaciones de escalamiento, traslación y rotación.

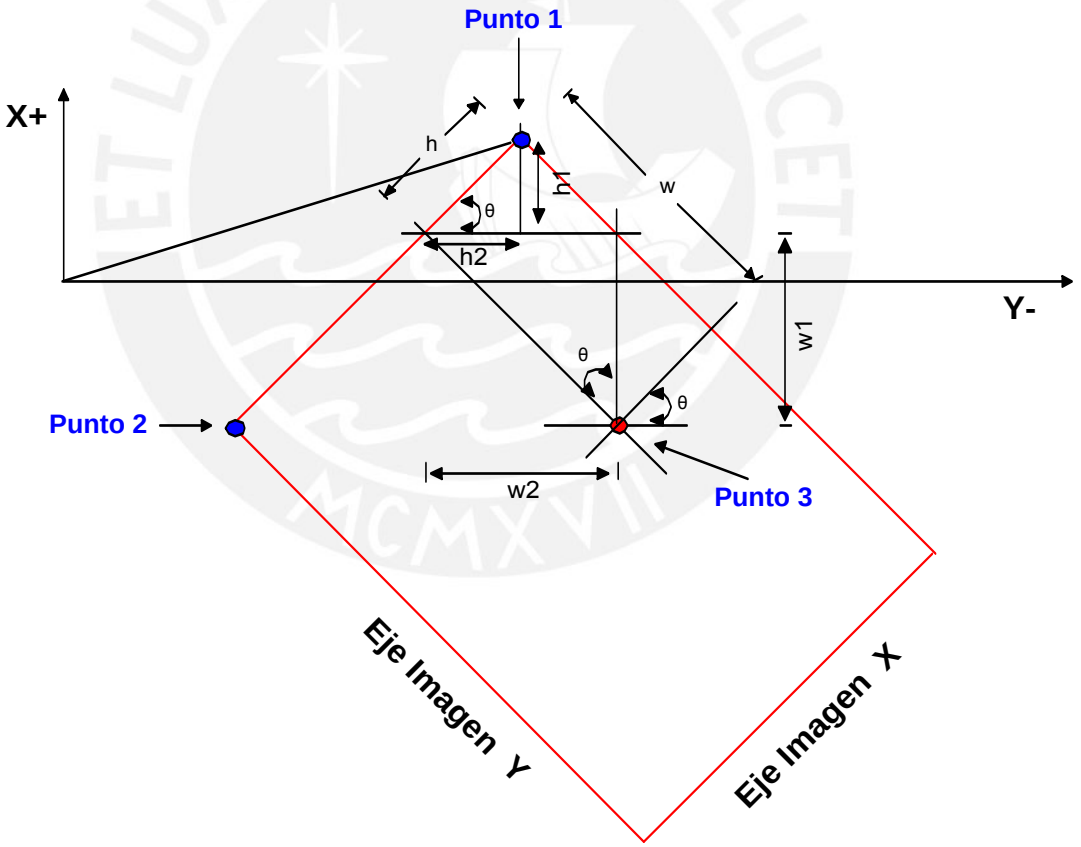



Figura 6.4 Transformación de Coordenadas

En la figura 6.4 podemos obeservar que:

$$h_1 = h\,\mathrm{sen}(\theta)$$
$$h_2 = h\cos(\theta) \qquad (6.1)$$

$$w_1 = w\cos(\theta)$$
$$w_2 = w\,\mathrm{sen}(\theta) \tag{6.2}$$

Podemos notar que conociendo las coordenadas del punto 1 y punto 2 podemos obtener el ángulo de rotación del frame. Si $(x_1,y_1)$ e $(x_2,y_2)$ son las cordenadas del punto 1 y punto 2 respectivamente entonces tenemos:

$$\mathrm{tag}\theta = \frac{x_1 - x_2}{y_1 - y_2} \tag{6.3}$$

$$\theta = \mathrm{atag}\left(\frac{x_1 - x_2}{y_1 - y_2}\right) \tag{6.4}$$

donde $\theta$ nos representa el ángulo de rotación buscado.

Nosotros conocemos además las coordenadas del objeto (punto 3) en coordenadas imagen (w,h), entonces la idea es transformar dichas coordenadas a las coordenadas cartesianas X,Y del robot, por tanto observando en la figura 6.4 tenemos que las coordenadas en el punto 3 es:

$$x_3 = h_1 + w_1$$
$$y_3 = w_2 - h_2 \tag{6.5}$$

entonces, reemplazando finalmente tenemos las coordenadas deseadas:

$$x_3 = x_1 + h\,\mathrm{sen}\theta + w\cos\theta$$
$$y_3 = x_1 + w\,\mathrm{sen}\theta - h\cos\theta \tag{6.6}$$

Como podemos observar se ha realizado una traslación y rotación de coordenadas. Notar que la coordenada imagen (w,h) esta dada en pixeles, por tanto previamente debe escalarse a milímetros, el cual es la unidad empleada en las coordenadas del robot.

## 6.6 Determinación de la orientación del objeto.

La idea es definir una dirección única de referencia del objeto. Esta dirección de referencia va a ser el denominado eje de mínima inercia, que obviamente es el mismo para un objeto bidimensional especifico. Así en la figura 6.5 el problema es calcular el ángulo que forma el eje de mínima inercia y uno de los ejes coordenados de la imagen (en la figura 6.5 el eje y)

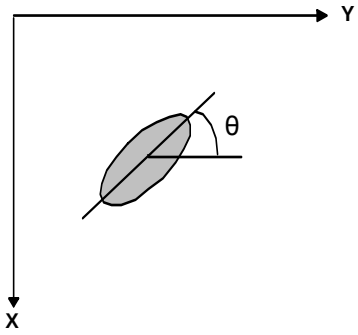



Figura 6.5 Objeto girado un ángulo θ

entonces la ecuación de una recta que pasa por un punto genérico (a, b) forma un ángulo θ con el eje y viene dada por:

$$(y-b)\cos\theta = (x-a)\operatorname{sen}\theta \tag{6.7}$$

el momento de inercia de un sólido bidimensional (el objeto) respecto a la recta anterior es por definición:

$$I = \sum_x \sum_y [(x-a)\operatorname{sen}\theta - (y-b)\cos\theta]^2 f(x,y) \tag{6.8}$$

siendo f(x,y) la función definida por el sólido. Obviamente, particularizando esta definición al caso que nos ocupa, la función f(x,y) será aquí la función binaria Io(x,y) definida por el objeto.

Para minimizar el momento de inercia, se deberá resolver el sistema de dos ecuaciones:

$$\frac{\partial I}{\partial x} = 0; \frac{\partial I}{\partial y} = 0 \tag{6.9}$$

desarrollando se llegaría a:

$$m_{10}\operatorname{sen}\theta - am_{00}\operatorname{sen}\theta - m_{01}\cos\theta + bm_{00}\cos\theta = 0 \tag{6.10}$$

que en virtud de la definición de centro de gravedad se puede reescribir como:

$$(\bar{x}-a)\operatorname{sen}\theta - (\bar{y}-b)\cos\theta = 0 \tag{6.11}$$

de lo cual se deduce que la recta de mínima inercia pasa por el centro de gravedad $(\bar{x},\bar{y})$ del objeto, ya que cumple la ecuación de la recta dada por la expresión 6.8:

$$\begin{aligned} \bar{x} &= a \\ \bar{y} &= b \end{aligned} \tag{6.12}$$

para obtener el ángulo θ se aplicaría la condición del mínimo:

$$\frac{\partial I}{\partial \theta}(\bar{x},\bar{y}) = 0 \tag{6.13}$$

obteniéndose la ecuación:

$$tg^2\theta + (\mu_{20} - \mu_{02})tg\theta/\mu_{11} + 1 = 0 \tag{6.14}$$

dando lugar al ángulo θ tal que:

$$tg2\theta = 2\mu_{11}/(\mu_{20} - \mu_{02}) \qquad (6.15)$$

una vez obtenido la dirección de referencia, entonces habría que girar la pinza del robot un ángulo $\theta$.

# Capítulo 7

# Implementación del Sistema de VA

## 7.1 Introducción

A continuación vamos a describir el diseño e implementación del sistema de Visión Artificial que nos permita seleccionar piezas mediante un brazo robótico.

En el desarrollo de este capítulo se describirá el hardware empleado y además las partes más importantes y sobresalientes del código de los programas de cada etapa del sistema, de manera que se pueda entender claramente el desarrollo del software.

## 7.2 Implementación del Hardware

### 7.2.1 Cámara y Tarjeta de Captura de Video

Se empleo una minicamara a color con sensores CCD *transfer interline* (ver capítulo 2) y hemos utilizado una tarjeta de captura de video multifunción, TV Tuner/Video Capture/FM, basado en el chip BT878 el cual es una completa solución para capturar señales sobre el Bus PCI, como se observa en la figura 7.1. Dicha tarjeta se usa comúnmente para visualizar canales de TV y recibir señales de radio desde una PC.

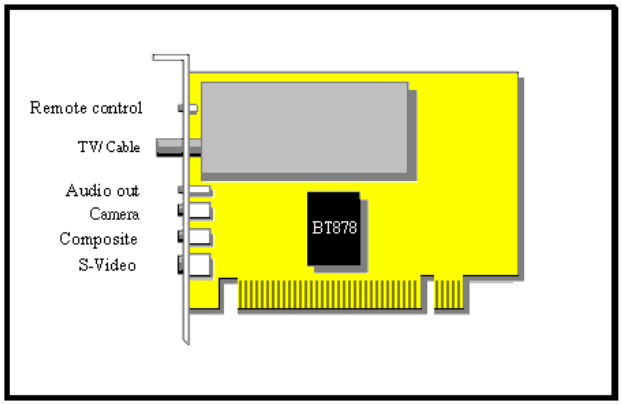



Figura 7.1 Tarjeta PCI digitalizadora de Video

El Bt838 coloca el dato de video directamente en la memoria central de la computadora. Como se puede tomar control del bus PCI tan pronto como esté disponible, entonces se evita la necesidad de tener buffers de almacenamiento de cuadro (imagen) incorporado.

### 7.2.2 Interface PC – Robot Scorbot

El software de visión artificial se desarrolla en una PC Pentium III, la cual contiene la cámara de video que captura la imagen de los objetos, para luego llevar la señal de video a una tarjeta digitalizadora de imágenes. Esta computadora deberá enviar las consignas al robot, como resultado del análisis realizado, dichas consignas son la información necesaria (posición de los objetos, orientación, numero de objetos, etc) para que el robot realice la tarea determinada y se envía mediante una conexión serial entre el controlador del robot y la PC.

Para ello se utilizo el canal de comunicación 1 del controlador del robot y el COM1 de la computadora como se muestra en la figura 7.2.

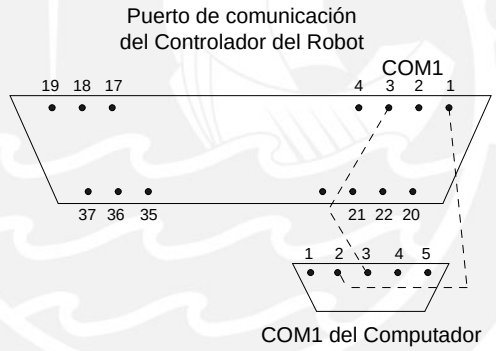



Figura 7.2 Enlace serial entre Controlador del Robot y PC Visión

### 7.2.3 Sistema de iluminación

Como mencionamos anteriormente, la iluminación es un factor muy importante en el proceso de visión para la obtención de resultados óptimos. Una deficiente iluminación implica obtener imágenes de baja calidad, lo que a su vez conlleva a resultados erróneos o un procesamiento excesivo.

Inicialmente observemos la figura 7.3, donde se puede apreciar una imagen sin ningún tipo de iluminación. Claramente se observa que los objetos son detectados fácilmente; sin embargo se presentaran problemas cuando consideremos las condiciones reales de funcionamiento de la celda de manufactura durante todo el día.

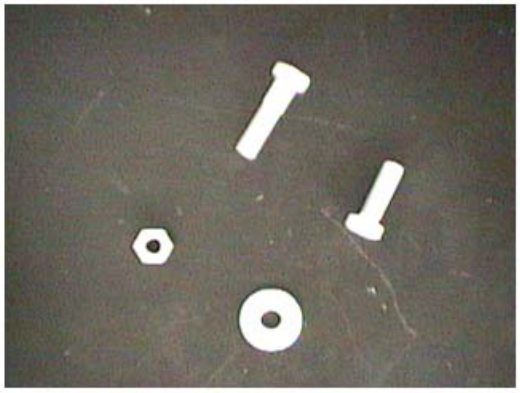


Figura 7.3 Escena sin iluminación

En la decisión del sistema de iluminación se ha considerado factores como:

a. *La ubicación del sistema robótico:* puesto que un lado de la mesa es iluminado por luz natural a través de una ventanas, y por el lado contrario la mesa presenta una zona más oscura.
b. *La cámara:* aprovechamos la opción de control automático de ganancia (AGC) el cual permite disminuir los excesivos brillos producidos tanto en los objetos como en la mesa.
c. *La iluminación ambiental e iluminación frontal fija*: el ambiente del CETAM tiene una iluminación propia (luz blanca potente en el techo) e iluminación natural (por las ventanas) que producen una iluminación ligeramente variable y no adecuada en ciertos momentos.
d. *La hora del día:* la iluminación es variable durante el día.
e. *Las posibles reflexiones:* existentes en la mesa o sobre la superficie del objeto.

Observando dichos problemas, y luego de una serie de ensayos, se decidió aprovechar parte de la iluminación que se encuentra instalada en el CETAM: ***lamparas con luz blanca*** usando la disposición de ***iluminación frontal*** como se muestra en la figura 7.4.

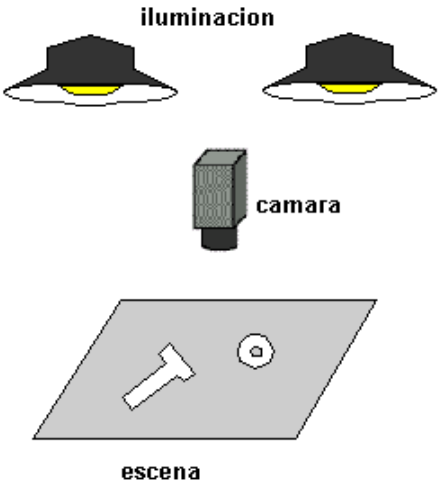



Figura 7.4 Sistema de iluminación frontal empleada

Figura 7.5 Escena con el sistema de iluminación frontal

Como observamos en la figura 7.5 se obtiene ciertos brillos en la superficie de los objetos pero que no es problema para nuestros objetivos, pues interesa más la forma (contorno) que la superficie interna del objeto.

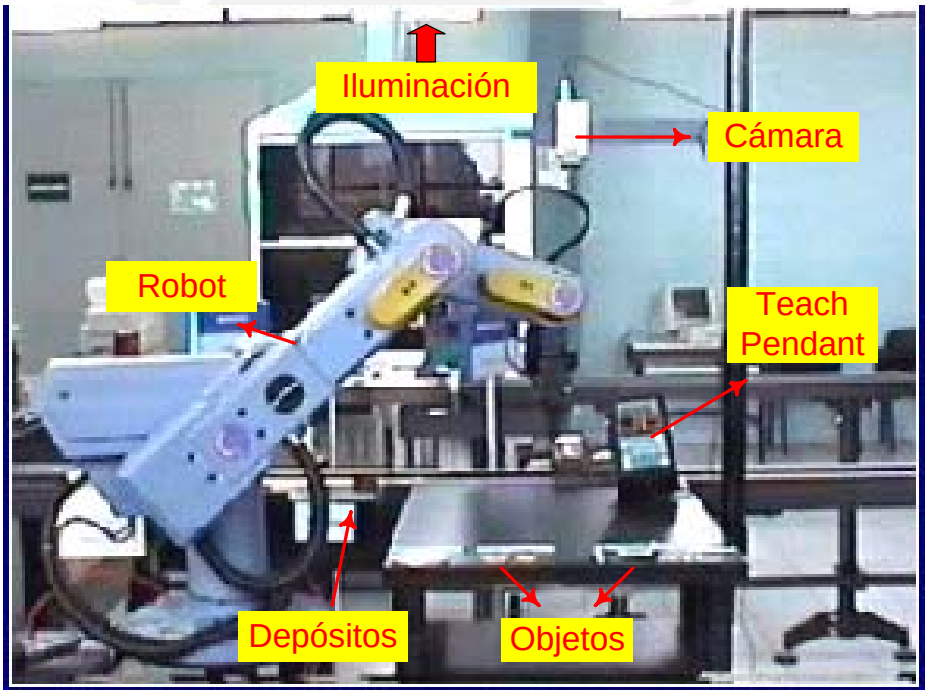
Sobre la superficie de la mesa se encuentra ciertos brillos pero que son regulados con el AGC de la cámara. Además se observa que la parte central de la escena recibe mas iluminación que los bordes, esto es una características del tipo de iluminación frontal la cual no es perjudicial para poder detectar el contorno de los objetos.

Es importante mencionar que si bien es cierto que no se consigue una iluminación optima, sin embargo es aceptable para nuestros propósitos, además lo mejor es

instalar una estructura de ***iluminación direccional*** alrededor de la escena, de manera que se ilumine en forma homogénea la totalidad de la escena sin producir brillos perjudiciales durante todo el día.

Para completar los componentes hardware que forman parte del sistema de visión implementado mencionaremos al manipulador Scorbot ER-IX con su controlador respectivo, mesa donde se coloca las piezas a reconocer, PC Pentium III donde se implementa el software de visión y otra que sirve como terminal para programar al robot.

Finalmente en la figura 7.6 se observa el sistema completo implementado, donde se muestra, en lo posible, cada uno de los componentes hardware empleado.



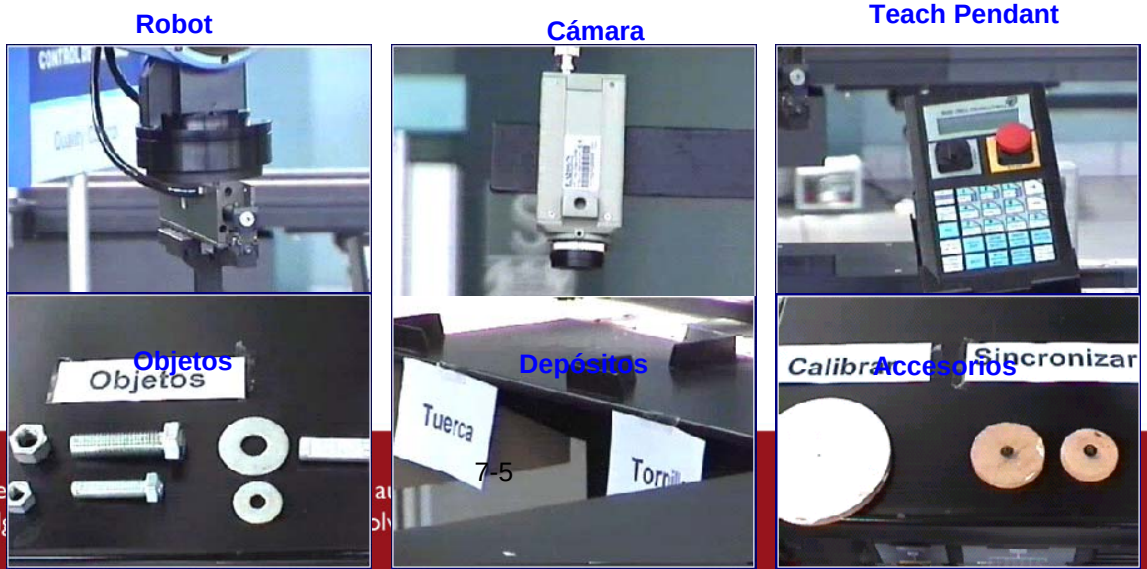



Figura 7.6 Componentes del Sistema Implementado

## 7.3 Implementación del Software

Se ha seguido la estrategia de desarrollo que consistió en emplear Matlab en una primera fase de desarrollo de los programas. La ventaja de la implementación en Matlab es la facilidad para poder realizar cambios en una primera etapa y poder diseñar los algoritmos correspondientes para nuestra aplicación y así posteriormente implementarlos en Visual C, que es el lenguaje en el que se implementó finalmente el sistema de VA. Por tanto en algunas partes de la explicación se emplea el código desarrollado en Matlab ya que el desarrollado en C es mas extenso.

### 7.3.1 Captura de la Imagen

Para poder obtener la imagen capturada por la cámara es necesario contar con un programa (driver) que acceda los datos (imagen) desde la tarjeta de digitalización de imágenes. Sin embargo el driver viene ya con la compra de la tarjeta y afortunadamente en visual C existe un programa el cual se enlaza al driver de la tarjeta de imagen que proporciona el proveedor de la tarjeta de imágenes, de tal manera que el objetivo inicial se centró en interpretar dicho programa y obtener la imagen como una matriz.

Cada elemento de la matriz corresponde a un pixel de la imagen, y en nuestro caso se tienen tres matrices, una para cada combinación de rojo, verde y azul de la imagen (RGB). Puesto que dicho programa es demasiado extenso, parte de ello se describe en el apéndice A.

### 7.3.2 Preprocesamiento y Segmentación de la imagen

Inicialmente la imagen puede estar degradada para nuestros objetivos, debido a causas aleatorias en los procesos de captación, transmisión y digitalización de la imagen, así como de iluminación. Por ello, en esta fase se pretenden eliminar estas distorsiones o mejorar las características de la imagen original para su posterior tratamiento.

Para la elaboración de este proyecto se analizó una serie de imágenes digitales, a las cuales debían aplicarse diversas técnicas de procesamiento de imagen y ver cuales de ellas serian implementadas.

El objetivo final de esta etapa es separar al objeto que se quiere inspeccionar del fondo y de los alrededores para posteriormente extraer características del objeto dentro de la imagen.

A continuación se muestran los pasos realizados en la etapa del preprocesamiento y segmentación de la imagen, basados en los algoritmos analizados en la sección 2.3-2.4.

La secuencia que se siguió en esta etapa se muestra a continuación:

a. Conversión a niveles de gris.
b. Filtrado
c. Umbralización.
d. Erosión y dilatación.
e. Etiquetado de la imagen binarizada.
f. Eliminación del ruido de binarización.

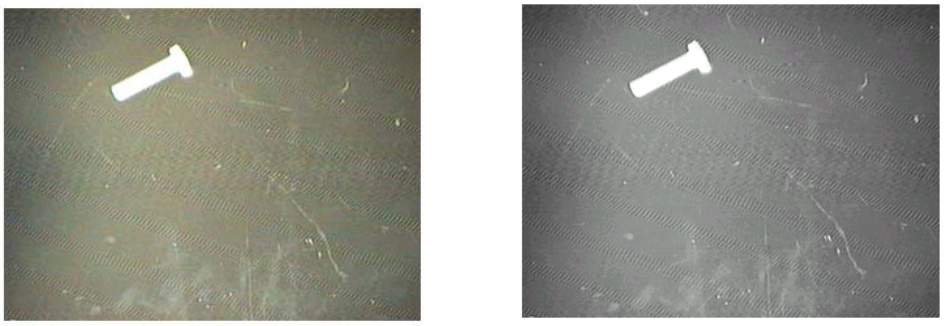
#### 7.3.2.1 Conversión a niveles de grises

Como se trabajo con una cámara de color, entonces el primer paso consiste en convertir la imagen a niveles de gris para tener pixeles con niveles de intensidad entre 0 y 255. Para ello se uso la ecuación mencionada en la capitulo 2:

$$Y = R * 0.3 + G * 0.5 + B * 0.11$$

```
for( k=0,i=0,j=0; k <76800;k++)
   {
      if((lpVHdr->dwBufferLength) < 76800*3)
         break;
         buffPixel[k].byte1=*pByte++;  //obtengo Blue
         buffPixel[k].byte2=*pByte++;  //obtengo Green
         buffPixel[k].byte3=*pByte++;  //obtengo Red
         //convierto la imagen de color a niveles de gris
         tempo=buffPixel[k].byte1*0.11 + buffPixel[k].byte2*0.59
         + buffPixel[k].byte3*0.3;
         imagen1[0][j++]=tempo;
   }
```

Figura 7.7 Conversión a niveles de grises



(a) (b)

Figura 7.8 (a) Imagen a color (b) Imagen en niveles de gris

#### 7.3.2.2 Filtrado

Inicialmente se aplico filtros para eliminar ruidos y mejorar la imagen, sin embargo en la mayoría de los casos cuando la imagen es poco ruidosa, el no emplear un filtro no afecta al momento de segmentar, es por ello que para evitar mayor tiempo de procesamiento se puede prescindir de un filtro, y se evitó su uso en la implementación final.

#### 7.3.2.3 Umbralización

Dada la variación de las características de luminosidad de las imágenes durante el día, no es posible determinar un umbral único para la imagen que se desea procesar.

Para separar un objeto del fondo de la imagen, se analiza el Histograma para calcular los límites de los niveles de escalas de grises (límites de corte) a partir de los cuáles se define el objeto que está siendo analizado.

La figura 7.9 del histograma nos indica que el fondo (gris oscuro) es lo que ocupa más espacio en la imagen, porque es el que contiene mayor número de pixels en la escala de grises; luego tenemos la pieza (gris claro, casi blanco). En nuestro caso se escogió un umbral (threshold) de 200.

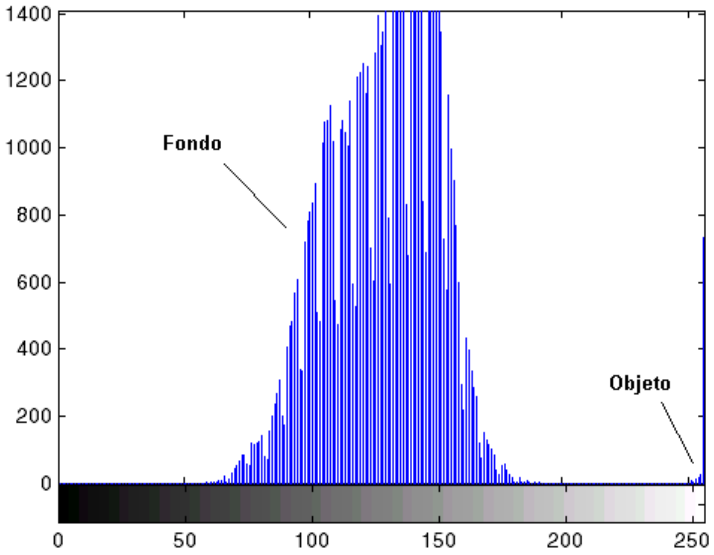



Figura 7.9 Histograma de la Imagen del Tornillo

Realmente lo que se hace es convertir la imagen de diferentes escalas de grises, en una imagen binaria (blanco y negro), en la cuál la imagen pasa a tener todos los pixeles que son menores al límite especificado en el threshold a un valor de cero (0), y todos los demás a un valor de uno (1).

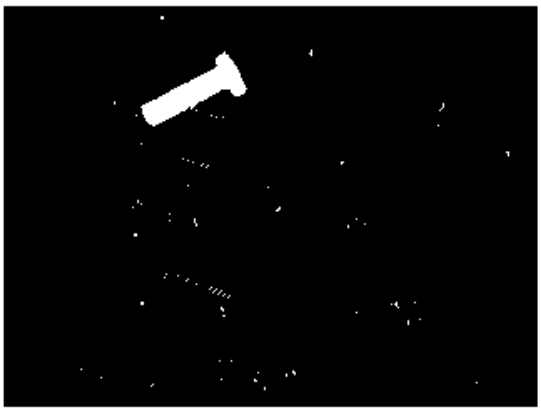


Figura 7.10 Imagen Binarizada del Tornillo

Es importante mencionar que la imagen binaria que resulta del threshold ocupa mucho menos memoria que una imagen de escala de grises o de color, pero contiene

la información de relevancia. Además se puede observar que aparece ciertos ruido en algunas zonas, el cual se llama ruido de binarización.

El código del algoritmo de umbralización (segmentación) es el siguiente

```
        for(y=1;y<resol_y+1;y++)                //recorre filas
            {
            for(x=1;x<resol_x+1;x++)            //recorre columnas
                  {
                  if(data[y][x]>= umbral1)
                      data[y][x]=1;
                  else
                      data[y][x]=0;
                  }
            }
```

Figura 7.11 Umbralización

#### 7.3.2.4 Erosión y Dilatación

Estas funciones se encargan básicamente de eliminar el ruido de conversión binaria, con una erosión del objeto seguido de la dilatación.

Luego de la erosión y dilación aplicada, el contorno del objeto original se volverá un poco fallido, pero normalmente no tendrá importancia; después de todo, el contorno total del objeto binario será básicamente conservado y las indeseadas fluctuaciones de pixel en la imagen serán eliminadas.

El elemento estructurador tanto para la erosión y dilatación, es un entorno cuadrado de 3x3, el cual, permite disminuir el tamaño de los fragmentos de ruido. La eliminación de los pixeles indeseados no es total, razón por lo cual, es necesario iterar la erosión, más de una de vez. Además con el fin de mantener la forma original del objeto, la dilatación se aplica el mismo número de iteraciones que la erosión.

El código del programa asi como su resultado se muestra a continuación:

```
Void erosion(void)
{    for(x=1;x<resol_x+1;x++)            //recorre columnas
      {
      for(y=1;y<resol_y+1;y++)      //recorre filas
            {
      if(data[y][x])          //si encuentro pixel=1-> erosiono
      {
     if(data[y-1][x-1]&&data[y][x-1]&&data[y+1][x-1]&&data[y-
     1][x]&&data[y][x]&&data[y+1][x]&&data[y-
      1][x+1]&&data[y][x+1]&&data[y+1][x+1])
        temp[y][x]=1;
      else
      temp[y][x]=0;
```

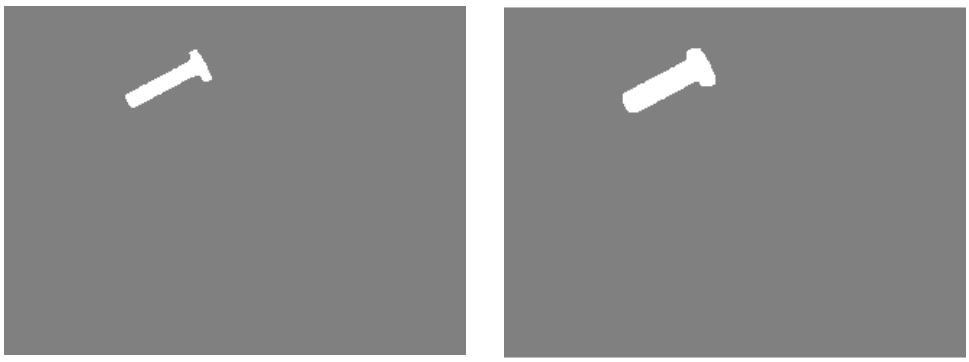
```
            }
      }
}

Void dilatacion(void)
{     for(x=1;x<resol_x+1;x++)      //recorre columnas
            {
      for(y=1;y<resol_y+1;y++)     //recorre filas
                  {
      if(data[y][x])          //si encuentro pixel=1-> dilato
      {     temp[y-1][x-1]=1;
            temp[y][x-1]=1;
            temp[y+1][x-1]=1;
            temp[y-1][x]=1;
            temp[y][x]=1;
            temp[y+1][x]=1;

            temp[y-1][x+1]=1;
            temp[y][x+1]=1;
            temp[y+1][x+1]=1;
            }
      }
   }
}
```

Figura 7.12 Funciones de erosión y dilatación

Como podemos observar en la figura 7.13, luego de aplicar operaciones morfológicas, en la imagen sólo aparece el objeto dentro del fondo y ya no existe otras regiones con pixeles blancos, aunque podría existir aún algunas áreas blancas como se dio en muchas ocasiones. Por tanto para evitar que ello ocurra y que el sistema sea más robusto es necesario algoritmos de etiquetado y filtrado como se verá a continuación.



(a) (b)

Figura 7.13 (a) Imagen erosionada (b) Imagen dilatada

#### 7.3.2.5 Etiquetado de objetos en la imagen

Luego de la etapa morfológica no podemos asegurar que ya no existe algún elemento irrelevante dentro de la imagen, es por eso que hay que etiquetar todos los elementos presentes dentro de la imagen binaria.

En esta etapa, se hace un barrido de la imagen desde la parte superior izquierda hacia la parte inferior derecha, y se registra todos los 'objetos' posibles. Cada uno de estos 'objetos' identificados se registra bajo un "numero" particular. Todos los pixeles del primer objeto que aparezcan en la parte superior izquierda serán etiquetados con 1, los del segundo con 2 y así sucesivamente. Por tanto, se pueden identificar varios objetos por ventana, numerados y analizados por separado.

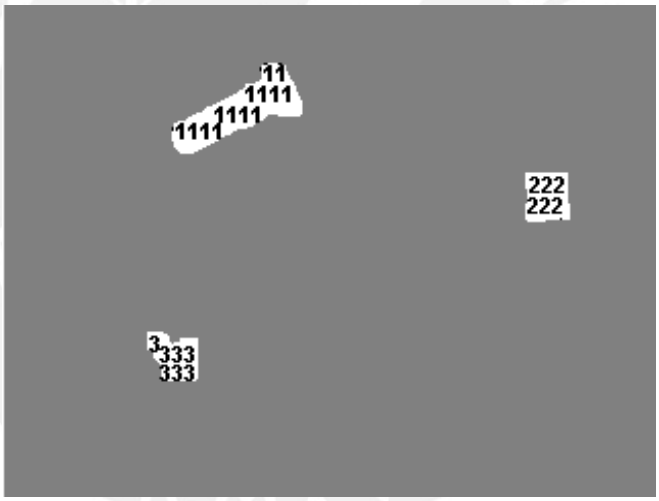



Figura 7.14 Imagen etiquetada

En la figura 7.14 se observa los 'objetos' cuyos pixeles pertenecientes han sido etiquetadas con un número. Pero observemos que realmente hay un objeto de interés (tornillo) y los demás son solo acumulación de pixeles blancos(ruido) que por algún motivo no pudieron eliminarse aún con la etapa de morfología. Por tanto en estos casos es necesario aplicar un filtro de tamaño. A continuación se muestra el código del programa de etiquetado.

```
for(y=1;y<resol_y+1;y++)           //recorro filas de data[][]
{
 for(x=1;x<resol_x+1;x++)     //recorro columnas de data[][]
 {if(data[y][x])  //busco pixel=1 en data[][]
```

```
{
if((temp[y][x-1]==0)&&(temp[y-1][x-1]==0)&&(temp[y-1][x]==0)
    &&(temp[y-1][x+1]==0))
     {N=N+1;                  //incremento si no hay vecinos
     temp[y][x]=N;}           //etiqueto nuevo pixel en temp[][]
     else
     {max1=max(temp[y-1][x-1],temp[y-1][x]);
      max2=max(temp[y-1][x+1],temp[y][x-1]);
      temp[y][x]= max(max1,max2);
      tabla[temp[y][x]][temp[y-1][x-1]] =1;
      tabla[temp[y][x]][temp[y-1][x] ] =1;
      tabla[temp[y][x]][temp[y-1][x+1]]=1;
      tabla[temp[y][x]][temp[y][x-1]  ]=1;}
           } //if()
     } //for() de x
     } //for() de y
```

Figura 7.15 Etiquetado de Objetos + Ruido con # 1,2,3…

#### 7.3.2.6 Filtro de tamaño

Hemos visto que la aplicación de operaciones morfológicas no siempre asegura que solo lo que queda en la imagen son objetos validos, pues se puede dar el caso que exista pequeñas acumulaciones de pixeles blancos llamados blobs, que no pertenecen a ningún objeto. Entonces nosotros debemos tener idea estimada la cantidad de pixeles que ocupa los objetos de nuestro interés, de tal manera que todos los 'objetos' que tengan mucho menor tamaño que el que fijemos se deberán filtrar.

En la figura 7.16 observamos que ahora sí queda solamente en la imagen el objeto de nuestro interés luego de aplicar el filtro de tamaño. A continuación se lista el programa que se encarga de filtrar acumulación de pixeles.


Figura 7.16 Resultado de eliminar elementos menor que un tamaño

```
for (j=1;j<et_final+1;j++)   //numero de veces =  # objetos
 {
 if(etiqueta2[1][j]<=umbral2)//si cumple --> borramos pixels
```

```
 {
for(y=1;y<resol_y+1;y++)          //recorro filas de data[][]
{for(x=1;x<resol_x+1;x++)    //recorro columnas de data[][]
 {
if( data[y][x]==j) //busca el numero j para reemplazar con cero
{data[y][x]=0;}
  }//for() de x
  }//for() de y
 etiqueta2[1][j]=0; //ver que etiqueta se elimino y cuales quedan,
  }//if()  //informacion es importante para el codigo de cadena
 }
```

Figura 7.17 Eliminación de pixeles aislados (Filtro de Tamaño)

### 7.3.3 Cálculo de las Características

Una vez obtenida la imagen libre de ruido y conteniendo sólo el objeto o los objetos de nuestro interés, se procede a extraer las características de cada objeto.

Como vimos, aunque existen diversos métodos para la selección de características, en el presente trabajo primero se usa la codificación del contorno basado en el código de cadena, para luego a partir de allí poder calcular algunos descriptores basado en momentos invariantes a rotaciones, traslaciones y homotecias, el cual fue desarrollado en el Capítulo 3.

Entre las características seleccionadas tenemos el conjunto de 4 de los 7 momentos invariantes para patrones de entrada a la red neuronal. Se ha visto que para aplicaciones practicas estos 4 valores de los momentos son suficientes para diferenciar entre los patrones de entrada.

Los momentos de igual orden, para cada uno de los 4 posibles objetos, respectivamente, proporcionará una medida de la similitud entre los objetos. Como los momentos son invariantes, todos los objetos del mismo tipo que se diferencien por su tamaño, posición y ángulos de rotación relativa son vistos como objetos iguales. Por tanto, un objeto almacenado con un determinado tamaño o posición será reconocido e identificado con cualquier otro tamaño o posición. En la figura 7.18 vemos los 4 posibles objetos a reconocer.


Figura 7.18 Objetos a Reconocer

Básicamente, los ordenes mas bajos de los momentos de inercia definen el objeto macroscópicamente con valores mayores, mientras que los ordenes superiores se ocupan de la parte microscópica, refinando las líneas.

#### 7.3.3.1 Cálculo del código de cadena de un objeto

Como se menciona en el Capítulo 3, la idea es codificar los bordes de los objetos, para ello se hace uso de un tipo de descriptor, el cual es el llamado código de cadena y así poder trabajar posteriormente en base al contorno y no al interior del objeto, lo cual es beneficioso ya que el procesamiento de la información se reduce considerablemente. El código del programa que realiza la codificación de cadena del objeto a analizar se muestra en la figura 7.19.

```
while( p==0)
     p=M(i,j);              %obtengo un pixel nuevo
     if (p==1)              %verifica si pixel es 1
      break
     elseif (j>=jmax & i>=imax)  %verifica ultimo pixel
      disp('no se encontro imagen');
     break;
     elseif (j>=jmax)       %verifica ultimo columna
          j=1;              %columna 1
          i=i+1;            %incremento fila
      end
     j=j+1;                 %incremento columna
end
%-------------------------------------------------------------
pix_ini=[i,j];              %coordenada inicial encontrada
i_ini=i;                    %almaceno las coordenadas iniciales
```

```
j_ini=j;                    %almaceno las coordenadas iniciales
m=1;                        %indices (fila) de la matriz del codigo cadenas
n=1;                        %indices  (columna)de la matriz del cod.cadenas
%-----------------------------------------------------------
inew=0;
jnew=0;
x=1;                     %fila de las cordenadas
p=M(i,j);             %valor de pixel inicial
while(~(inew ==i_ini & jnew ==j_ini))
a=M(i-1,j);             %obtengo valor de pixel
b=M(i-1,j+1);        %obtengo valor de pixel
c=M(i,j+1);             %obtengo valor de pixel
d=M(i+1,j+1);        %obtengo valor de pixel
e=M(i+1,j);             %obtengo valor de pixel
f=M(i+1,j-1);        %obtengo valor de pixel
g=M(i,j-1);             %obtengo valor de pixel
h=M(i-1,j-1);        %obtengo valor de pixel

if (b==1 & a==0)    %comparo si pixel anterior es 0
     cadena(m,n)=1; %codigo 1
     n=n+1;
     i=i-1; j=j+1;  %actualizo coordenadas de b
elseif (c==1 & b==0)    %verifica ultimo pixel
       cadena(m,n)=0;   %codigo 0
       n=n+1;
       i=i; j=j+1;      %actualizo coordenadas de c
elseif (d==1 & c==0)    %verifica ultimo pixel
       cadena(m,n)=7;   %codigo 7
       n=n+1;
      i=i+1; j=j+1;    %actualizo coordenadas de d
elseif(e==1 & d==0) %verifica ultimo pixel
       cadena(m,n)=6;   %codigo 6
       n=n+1;
      i=i+1; j=j;       %actualizo coordenadas de e
elseif (f==1 & e==0)    %verifica ultimo pixel
       cadena(m,n)=5;   %codigo 5
       n=n+1;
      i=i+1; j=j-1;    %actualizo coordenadas de f
elseif (g==1 & f==0)    %verifica ultimo pixel
       cadena(m,n)=4;   %codigo 4
       n=n+1;
      i=i; j=j-1;       %actualizo coordenadas g
elseif (h==1 & g==0)    %verifica ultimo pixel
       cadena(m,n)=3;   %codigo 3
       n=n+1;
       i=i-1; j=j-1;    %actualizo coordenadas de h
```

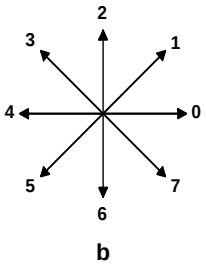
Figura 7.19 Cálculo el código de cadena para un objeto

```
elseif (a==1 & h==0)    %verifica ultimo pixel
       cadena(m,n)=2;   %codigo 2
       n=n+1;
       i=i-1; j=j;      %actualizo coordenadas de a
end  % final del if
inew=i;
jnew=j;
cordenadas(x,:)=[i j];  %almaceno coordenadas de la cadena
x=x+1;

end   %final del for o while
```

Figura 7.20 Cálculo el código de cadena para un objeto (continuación)

El resultado de dicho algoritmo aplicado a la imagen siguiente lo podemos observar en la figura 7.21. Notar que para la codificación hemos utilizado una vecindad de 8




codigo_cadena: [ 0 0 0 0 0 0 7 0 6 6 7 6 6 7 7 7 7 6 7 0 0 0 6 7 7 0 7 6 7 7 7 6 7 7 7 6 7 7 7
7 7 7 6 7 7 7 6 7 7 6 6 6 6 5 5 5 5 5 5 4 4 4 4 4 4 2 3 3 4 2 3 3 3 3 3 2 3 3
3 3 2 3 3 3 3 3 2 3 3 3 2 3 2 2 2 3 3 2 3 3 4 4 3 3 3 2 2 2 2 2 2 2 2 2 1 1 0 1]

Figura 7.21 Codificación de cadena para un objeto (llavero) a reconocer

#### 7.3.3.2 Calculo de los Momento Invariantes

Una vez evaluado la codificación de cadena para el objeto a reconocer, podremos calcular los momentos en el siguiente orden:

- Momentos generales
- Momentos invariantes a traslaciones
- Momentos invariantes a homotecias, y finalmente
- Momentos de Hu invariantes a traslación, rotación y homotecia,

El algoritmo que calcula dichos momentos se presenta a continuación:

```
xo = cordenadas(t,1); %obtengo coordenadas iniciales de la cadena
yo = cordenadas(t,2); %obtengo coordenadas Y de la cadena
for I=1:t
x = cordenadas(I,1); %obtengo coordenadas X de la cadena
y = cordenadas(I,2); %obtengo coordenadas Y de la cadena
letra=cadena(1,I);   %obtengo codigo de la cadena
       if letra==0
           dx= 0;
          dy= 1;
     elseif letra==1
           dx= -1;
           dy= 1;
     elseif letra==2
           dx= -1;
```

```
          dy= 0;
     elseif letra==3
          dx= -1;
          dy= -1;
     elseif letra==4
          dx= 0;
          dy= -1;
     elseif letra==5
          dx= 1;
          dy= -1;
     elseif letra==6
          dx= 1;
          dy= 0;
     elseif letra==7
          dx= 1;
          dy= 1;
     end
%-----------------------------------------------
if (dx==0)
   p=0;  q=0;
         foo = ( xo^(p+1) )*( y^(q+1)-yo^(q+1) );
         doo = doo + foo/((p+1)*(q+1));
   p=0;  q=1;
         fo1 = ( xo^(p+1) )*( y^(q+1)-yo^(q+1) );
         do1 = do1 + fo1/((p+1)*(q+1));
   p=1;  q=0;
         f1o = ( xo^(p+1) )*( y^(q+1)-yo^(q+1) );
         d1o = d1o + f1o/((p+1)*(q+1));
   p=1;  q=1;
         f11 = ( xo^(p+1) )*( y^(q+1)-yo^(q+1) );
         d11 = d11 + f11/((p+1)*(q+1));
   p=0;  q=2;
         fo2 = ( xo^(p+1) )*( y^(q+1)-yo^(q+1) );
         do2 = do2 + fo2/((p+1)*(q+1));
    p=2;  q=0;
         f2o = ( xo^(p+1) )*( y^(q+1)-yo^(q+1) );
         d2o = d2o + f2o/((p+1)*(q+1));
   p=1;  q=2;
         f12 = ( xo^(p+1) )*( y^(q+1)-yo^(q+1) );
         d12 = d12 + f12/((p+1)*(q+1));
   p=2;  q=1;
         f21 = ( xo^(p+1) )*( y^(q+1)-yo^(q+1) );
         d21 = d21 + f21/((p+1)*(q+1));
   p=0;  q=3;
         fo3 = ( xo^(p+1) )*( y^(q+1)-yo^(q+1) );
         do3 = do3 + fo3/((p+1)*(q+1));
p=3;  q=0;
          f3o = ( xo^(p+1) )*( y^(q+1)-yo^(q+1) );
          d3o = d3o + f3o/((p+1)*(q+1));
  yo = y; xo=x;
```

Figura 7.22 Cálculo de los momentos Invariantes

```
else
   pe=dy/dx;  %calculamos la pendiente
   p=0; q=0;
    foo = pe*(x^(p+2)- xo^(p+2));
   doo = doo + foo/((p+2)*(p+1));

 p=0; q=1;
 fo1=(pe/(p+3))*(x^(p+3)-xo^(p+3))+((yo-pe*xo)/(p+2))*(x^(p+2)- xo^(p+2));
 do1 = do1 + (pe)*(fo1/(p+1));
 p=1; q=0;
 f1o = pe*(x^(p+2)- xo^(p+2));
 d1o = d1o + f1o/((p+2)*(p+1));
 p=1; q=1;
 f11=(pe/(p+3))*(x^(p+3)-xo^(p+3))+((yo-pe*xo)/(p+2))*(x^(p+2)- xo^(p+2));
 d11 = d11 + (pe)*(f11/(p+1));
```

```
 p=1; q=2;
 f1 = (x^(p+4)- xo^(p+4))/(p+4);
 f2 = (x^(p+2)- xo^(p+2))/(p+2);
 f3 = (x^(p+3)- xo^(p+3))/(p+3);
 e  = yo-pe*xo;
 f12 = (pe^2)*f1 + (e^2)*f2 + 2*pe*e*f3;
 d12 = d12 + pe*f12/(p+1);
 p=2; q=1;
 f21=(pe/(p+3))*(x^(p+3)- xo^(p+3))+((yo-pe*xo)/(p+2))*(x^(p+2)-xo^(p+2));
 d21 = d21 + (pe)*(f21/(p+1));
 p=0; q=2;
 f1 = (x^(p+4)- xo^(p+4))/(p+4);
 f2 = (x^(p+2)- xo^(p+2))/(p+2);
 f3 = (x^(p+3)- xo^(p+3))/(p+3);
 e  = yo-pe*xo;
   fo2 = (pe^2)*f1 + (e^2)*f2 + 2*pe*e*f3;
   do2 = do2 + pe*fo2/(p+1);

   p=2; q=0;
    f2o = pe*(x^(p+2)- xo^(p+2));
   d2o = d2o + f2o/((p+2)*(p+1));
   p=0; q=3;
   f1 = (x^(p+5)- xo^(p+5))/(p+5);
    f2 = (x^(p+2)- xo^(p+2))/(p+2);
   f3 = (x^(p+4)- xo^(p+4))/(p+4);
   f4 = (x^(p+3)- xo^(p+3))/(p+3);
   e  = yo-pe*xo;
   fo3 = (pe^3)*f1 + (e^3)*f2 + 3*(pe^2)*e*f3 + 3*pe*(e^2)*f4;
   do3 = do3 + pe*fo3/(p+1);

   p=3; q=0;
    f3o = pe*(x^(p+2)- xo^(p+2));
   d3o = d3o + f3o/((p+2)*(p+1));
   yo=y;  xo=x;
    end
contador=contador+1;
end   %fin del for
xp=m1o/moo;     %cordenada x (vertical)
yp=mo1/moo;     %cordenada y (horizontal)
%calculo de momentos centrales invariantes a Traslaciones
%-------------------------------------------------------
uoo=moo;
uo1=0;
u1o=0;
u11= m11 - m1o*mo1/moo;
u2o= m2o - m1o^2/moo;
uo2= mo2 - mo1^2/moo;
u3o= m3o - 3*m1o*m2o/moo + 2*(m1o^3)/(moo^2);
uo3= mo3 - 3*mo1*mo2/moo + 2*(mo1^3)/(moo^2);
u21= m21 - (mo1*m2o/moo) - 2*m1o*m11/moo + (2*(mo1*m1o^2)/(moo^2));
u12= m12 - (m1o*mo2/moo) - 2*mo1*m11/moo + (2*(m1o*mo1^2)/(moo^2));
```

Figura 7. 23 Cálculo de los momentos Invariantes (Continuación)

```
%Normalizamos los momentos centrales para invarianza a Traslacion-Escala
noo = uoo/((uoo)^1.0);        %por defecto es: 1
n11 = u11/((uoo)^2.0);        %por defecto es: 0
n1o = u1o/((uoo)^1.5);        %por defecto es: 0
no1 = uo1/((uoo)^1.5);
n2o = u2o/((uoo)^2.0);
no2 = uo2/((uoo)^2.0);
n3o = u3o/((uoo)^2.5);
no3 = uo3/((uoo)^2.5);
n21 = u21/((uoo)^2.5);
n12 = u12/((uoo)^2.5);

%Evaluacion de Momentos Invariantes a Traslaciones-Rotaciones-Escala
phi1= n2o + no2;
```

```
phi2= (n2o - no2)^2 + 4*n11^2;
phi3= (n3o - 3*n12)^2 + (3*n21 - no3)^2 ;
phi4= (n3o + n12)^2 + (n21 + no3)^2 ;
phi5= (n3o - 3*n12)*(n3o + n12)*((n3o + n12)^2 - 3*(n21 + no3)^2)+ ...
(3*n21 - no3)*(n21 + no3)*(3*(n3o + n12)^2 - (n21 + no3)^2);
phi6= (n2o - no2)*((n3o + n12)^2 - (n21 + no3)^2) + ...
       4*n11*(n3o + n12)*(n21 + no3);
phi7= (3*n21 - no3)*(n3o + n12)*((n3o + n12)^2 - 3*(n21 + no3)^2) + ....
   (3*n12 - n3o)*(n21 + no3)*(3*(n3o + n12)^2 - (n21 + no3)^2);
```

Figura 7. 24 Cálculo de los momentos Invariantes (Continuación)

Para las siguientes imágenes de segmentos extraídos, se hallaron los siguientes Momentos invariantes (características):

| | **Tornillo Grande** | **Tornillo Trasladado** | **Tornillo Chico** | **Tornillo Rotado** |
|---|---|---|---|---|
| |  |  |  |  |
| Momentos Invariantes | Original(centro) | Traslación | escala | rotación |
| $\phi_1$ | 0.3699 | 0.3770 | 0.3772 | 0.3640 |
| $\phi_2$ | 0.0968 | 0.1074 | 0.1069 | 0.0979 |
| $\phi_3$ | 0.0047 | 0.0033 | 0.0024 | 0.0030 |
| $\phi_4$ | 0.0011 | 0.0011 | 0.0007 | 0.0011 |

Figura 7. 25 Momentos de inercia de Tornillos

De la misma manera se realizó para cada objeto diferente a reconocer, y estas características son las evaluadas para detectar un objeto.

# Capítulo 7

### 7.3.4 Diseño de la Red Neuronal

En esta parte, la idea principal radica en construir una Red Neuronal que permita clasificar los patrones en forma automática dentro de un número dado de clases. Para esto, era necesario encontrar características medibles que posibiliten la diferenciación de clases y que compitan con la fácil diferenciación visual que se puede realizar, como se vio en la etapa previa.

Entre las clases de redes neuronales que hay, los perceptrones multicapas con el algoritmo de Retropropagación del error han sido probados con éxito en la solución de varios tipos de problemas, y será el que emplearemos en éste trabajo.

El esquema operativo para la construcción de la red como clasificador constará de las siguientes partes:

#### 7.3.4.1 Construcción de una Base de datos

Como hemos comentado, el vector de características agrupa valores numéricos de características de cada pieza, que serán la entrada en la red neuronal y la categoría objetivo.

Se forma una pequeña base de datos el cual es un archivo que contiene 80 vectores característicos de los diferentes objetos ordenados aleatoriamente. A cada vector característico le corresponde una salida correspondiente el cual categoriza a un objeto determinado.

Los elementos del archivo se llaman pares de entrenamiento. Por ejemplo para el tornillo, un vector de entrada X (vector característico) y de salida Y, es:

$$X_1 = [0.3699 \quad 0.0968 \quad 0.0047 \quad 0.0011]^T$$

$$Y_1 = [0 \quad 0 \quad 0 \quad 1]^T$$

De manera similar para la tuerca, llavero y la arandela tenemos respectivamente:

$$X_1 = [0.1603 \quad 0.0000 \quad 0.0000 \quad 0.0000]^T$$

$$Y_1 = [0 \quad 0 \quad 1 \quad 0]^T$$

$$X_1 = [0.3261 \quad 0.0792 \quad 0.0005 \quad 0.0003]^T$$

$$Y_1 = [0 \quad 1 \quad 0 \quad 0]^T$$

$$X_1 = [0.1594 \quad 0.0000 \quad 0.0000 \quad 0.0000]^T$$

$$Y_1 = [1 \quad 0 \quad 0 \quad 0]^T$$

Por tanto el conjunto de entrenamiento para este problema es la agrupación de diferentes vectores característicos y está dado por el conjunto:

$$\{\{X_1, Y_1\}, \{X_2, Y_2\}, \{X_3, Y_3\}, \ldots\ldots \{X_{80}, Y_{80}\}\}$$

Nótese que la red neuronal requiere de una salida por cada clase, y si la red se usa como un clasificador, todas las componentes de cada vector salida son 0 excepto la componente (o componentes) que corresponde al patrón de entrada que se fija a 1.

Para el mejor desempeño de un sistema clasificador es posible añadir dos categorías a los tipos de cobertura que queremos clasificar, estas son: Confusos, Desconocidos. Estas dos categorías adicionales (aún no consideradas) liberan al sistema clasificador de dar respuestas concluyentes cuando existe confusión y evitan la excesiva generalización al clasificar los objetos que no corresponde a los objetos a reconocer.

#### 7.3.4.2 Separación de los datos

Del conjunto de patrones disponibles, se divide en un *conjunto de entrenamiento* y un *conjunto de validación*.

El conjunto de entrenamiento es generalmente mayor que el de validación, para el que basta con reservar del 10% al 30% de los datos disponibles, por ello utilizamos como conjunto de entrenamiento, 68 vectores de los 80 disponibles, reservando 12 vectores restantes como conjunto de prueba para realizar la etapa de reconocimiento:

Para el entrenamiento: entradas $(X_1, X_1, \ldots X_{68})$ y salidas $(Y_1, Y_1, \ldots Y_{68})$

Para la validación: entradas $(X_{69}, X_{70}, \ldots X_{80})$ y salidas $(Y_{69}, Y_{70}, \ldots Y_{80})$

Para el proceso de entrenamiento se crea un archivo que contiene los parámetros de entrada y la salida deseada de la red; luego se realizará una lectura del archivo y se ejecuta el aprendizaje hasta que el error de la red sea aceptable.

#### 7.3.4.3 Transformación de los datos de entrada

En este caso, los valores numéricos que componen el vector característico no es necesario que sea normalizado debido a que dichos valores son menores que la unidad.

#### 7.3.4.4 Arquitectura de la red Neuronal

La mejor arquitectura se escogió después de comparar el error de generalización para diferentes inicializaciones y cantidad de neuronas en la capa oculta.

El modelo de red neuronal artificial será ***perceptron multicapa*** cuya función de activación es la ***sigmoidal*** y utilizaremos el algoritmo ***backpropagation.***

El modelo de la red estará conformada por 4 capas: una capa de entrada, dos capas ocultas y una capa de salida.

En la capa de entrada el número de neuronas es igual al tamaño del vector de características producto de la caracterización de la imagen, que en este caso es igual a 4. Las capas ocultas constan de 8 neuronas cada capa como parámetro inicial pero pueden ser modificadas posteriormente.

Para la capa de salida, el número de neurones es igual al número de clases que desean ser reconocidas, en este caso es igual a 4 neurones (Tornillo, Tuerca, Arandela, Llavero).

La figura 7.26 muestra la arquitectura de la red neuronal a utilizar.

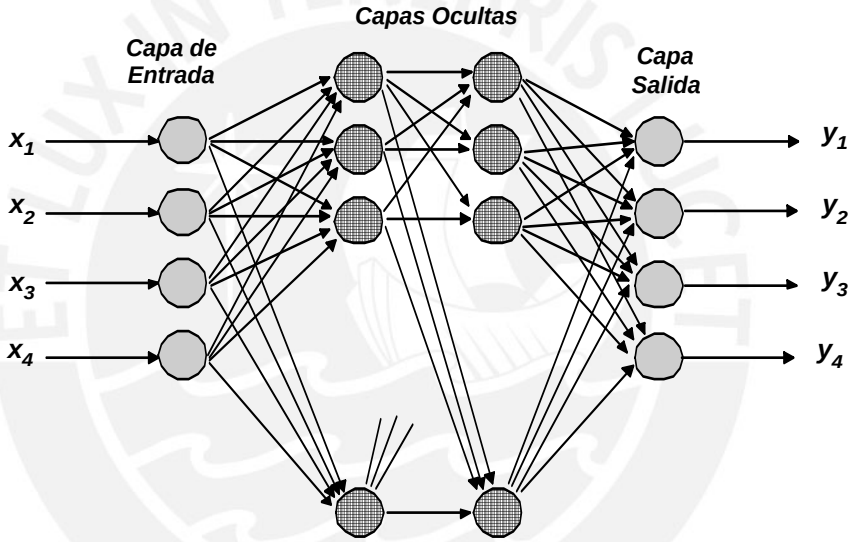



Figura 7.26 Estructura de la Red Neuronal utilizada.

#### 7.3.4.5 Fase de entrenamiento

Se evaluaron métodos de entrenamiento y se selecciono el método de Levenberg-Marquardt debido a su velocidad y estabilidad de convergencia

El entrenamiento realizado en nuestro caso será supervisado, ya que en la red neuronal empleada se utilizan patrones conocidos y ya clasificados y el sistema en consecuencia explota este conocimiento.

El tipo de entrenamiento que sea utilizado para el aprendizaje, será el algoritmo backpropagation.

Para realizar el entrenamiento de la red neuronal, utilizamos un coeficiente de inicial de aprendizaje de 0.01, además, los valores de las conexiones (W) entre cada uno de

los neurones (pesos), fueron inicializados con valores aleatorios. De manera similar, los umbrales también fueron inicializados con valores aleatorios (B).

Para realizar la etapa de entrenamiento del sistema utilizamos como conjunto de entrenamiento, 68 vectores de los 80 disponibles por cada sujeto, reservando 12 vectores restantes como conjunto de prueba para realizar la etapa de reconocimiento.

Durante las pruebas experimentales, se pudo evidenciar que si una red deja de aprender antes de llegar a una solución aceptable, se debe realizar un cambio en el número de neuronas ocultas, y volver a empezar el entrenamiento con un conjunto distinto de pesos.

- **Algoritmo de entrenamiento (Backpropagation)**

El algoritmo de entrenamiento por retropropagación, es un algoritmo iterativo por descenso del gradiente diseñado para minimizar el error cuadrático medio entre la salida real del Perceptrón Multicapa y la salida deseada.

Una vez pasado el algoritmo a todos los patrones del conjunto de entrenamiento, se calcula el error total correspondiente a esa iteración para el conjunto de entrenamiento. El proceso de iteración se repite hasta que el error total de una iteración es menor que una cantidad prefijada, o hasta que este se estabiliza. En este momento se salvan los pesos.

Es importante mencionar que el entrenamiento de la red se realizo en Matlab, cuyos pesos se almacenaron en un archivo y fueron usados posteriormente desde visual C.

A continuación se muestra el scrip de entrenamiento de la red

```
close all;clear all;
load patrones1;
X=x;
Yd=yd;
Error    = 1e-2;        % Mínimo Error deseado
P=X;     %P es el vector de patrones
T=Yd;    %T es el vector objetivo, ie la salida deseada
Ocultas=8;
[W1,B1,W2,B2,W3,B3] =
initff(P,ocultas,'tansig',ocultas,'tansig',T,'purelin');
[W1,B1,W2,B2,W3,B3,TE,TR] =
trainlm(W1,B1,'tansig',W2,B2,'tansig',W3,B3,'purelin',P,T,[10 3000
error]);
title('Entrenamiento de Backpropagation');
xlabel('Ciclos');
ylabel('Error Medio Cuadrático')
numero=numero_patrones;
x=x;
save pesos2.mat x yd X Yd W1 B1 W2 B2 W3 B3 TR error
numero_patrones;
```

Figura 7.27 Entrenamiento de la Red Neuronal

A continuación vemos las dimensiones de los parámetros de la red hallada:

| Nombre | Tamaño | Descripción |
|---|---|---|
| B1 | 8x1 | Bias 1 |
| B2 | 8x1 | Bias 2 |
| B3 | 4x1 | Bias 3 |
| TR | 1x1305 | Registro de entrenamiento (Numero de épocas) |
| W1 | 8x4 | Matriz de Pesos 1 |
| W2 | 8x8 | Matriz de Pesos 2 |
| W3 | 4x8 | Matriz de Pesos 3 |
| X | 4x68 | Matriz de patrones de entrada |
| Yd | 4x68 | Matriz de salidas deseadas |
| error | 1x1 | Error |

Tabla 7.1

En la figura 7.28 se muestra el comportamiento del entrenamiento de la red neuronal En esta gráfica podemos observar que el error total (Error Cuadrático Medio) obtenido por la red neuronal disminuye rápidamente durante los primeros instantes del entrenamiento. Luego se observa que el error total obtenido por la red neuronal varía muy poco con respecto al error total obtenido en épocas anteriores, sin embargo, el porcentaje de reconocimiento continúa mejorando a pesar de estas escasas variaciones en el error total.

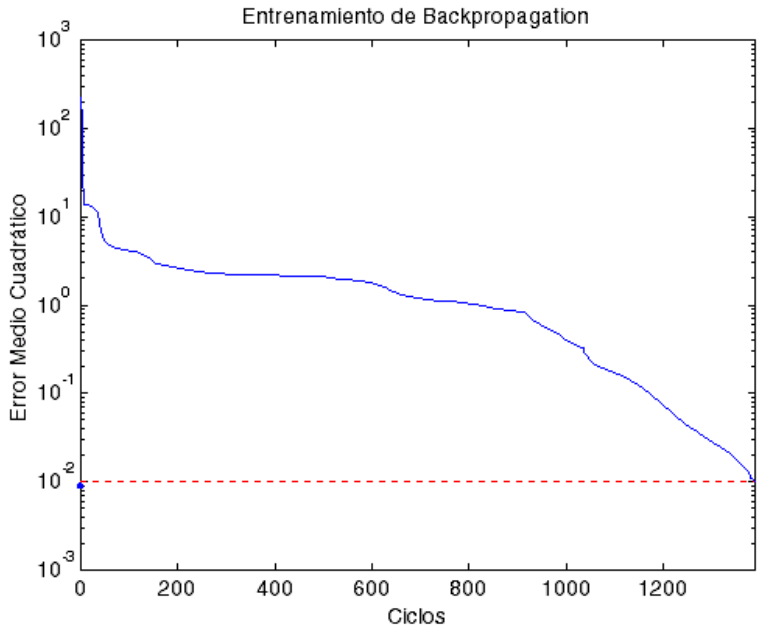



Figura 7.28 Comportamiento del entrenamiento de la Red Neuronal.

#### 7.3.4.6 Fase de Validación

Con los pesos obtenidos en el paso anterior se aplica a la red al conjunto de patrones que reservamos como conjunto de validación. Si el error cometido al aplicar la red al conjunto de validación es del orden del que obtuvimos al aplicar el algoritmo de entrenamiento al conjunto de patrones de entrenamiento, entonces se considera que la red tiene un comportamiento adecuado. Si no, sería necesario repetir el proceso variando la topología, la función de activación, el algoritmo de entrenamiento o el tipo de red.

Esto es una de las características más importantes de las redes neuronales: la generalización. No obstante, para evitar una excesiva generalización, recuerden que se puede introducir dos nuevas categorías: patrón desconocido y patrones confusos.

| Objeto | Salida deseada (yd) | | | | Salida Validada (y) | | | |
|---|---|---|---|---|---|---|---|---|
| Tuerca | 0 | 0 | 1 | 0 | 0.0050 | 0.0000 | 0.9949 | 0.0000 |
| Llavero | 0 | 1 | 0 | 0 | 0.0000 | 1.0000 | -0.0003 | 0.0002 |
| Arandela | 1 | 0 | 0 | 0 | 0.9953 | 0.0003 | 0.0043 | 0.0001 |
| Tornillo | 0 | 0 | 0 | 1 | -0.0001 | -0.0001 | 0.0000 | 1.0002 |
| Llavero | 0 | 1 | 0 | 0 | 0.0001 | 0.9997 | 0.0008 | -0.0005 |
| Tornillo | 0 | 0 | 0 | 1 | 0.0000 | -0.0001 | 0.0000 | 1.0001 |
| Tuerca | 0 | 0 | 1 | 0 | -0.0040 | 0.0002 | 1.0036 | 0.0001 |
| Tuerca | 0 | 0 | 1 | 0 | 0.0073 | -0.0002 | 0.9931 | -0.0001 |
| Arandela | 1 | 0 | 0 | 0 | 1.0174 | 0.0000 | -0.0173 | 0.0000 |
| Tornillo | 0 | 0 | 0 | 1 | 0.0000 | -0.0001 | 0.0000 | 1.0002 |
| Arandela | 1 | 0 | 0 | 0 | 1.0177 | 0.0003 | -0.0181 | 0.0001 |
| Arandela | 1 | 0 | 0 | 0 | 0.9982 | -0.0003 | 0.0024 | -0.0002 |
| Arandela | 1 | 0 | 0 | 0 | 0.9976 | 0.0000 | 0.0024 | 0.0000 |

Figura 7.29 Validación de la Red Neuronal

En donde el máximo error obtenido es: emax = 0.0181

Una vez generado y validado el clasificador podemos clasificar nuevos vectores de clase desconocida usando la red implementada.

### 7.3.5 Calibración y Sincronización Robot – Visión

En esta etapa desarrollaremos dos problemas muy importantes. El primero de ellos consiste en calcular las dimensiones en milímetros del marco de captura de la imagen; y la segunda es sincronizar el espacio de coordenadas de la imagen a las coordenadas del robot.

#### 7.3.5.1 Definiendo el Marco de la Imagen

Para determinar las dimensiones horizontal (Y) y vertical (X) en milímetros del marco de captura de la imagen, debemos realizar los siguientes pasos:

a. Colocar un objeto circular de diámetro conocido, aproximadamente en el centro del marco de captura.
b. Puesto que la mesa donde se colocara los objetos es de color negro, es recomendable que el objeto circular sea de color blanco.
c. Previamente, se debe realizar un análisis del histograma de la imagen capturada, y hallar el umbral óptimo de tal manera que se elimine ruido de fondo, manchas, suciedad, etc sin afectar demasiado los pixeles que pertenecen al objeto de interés

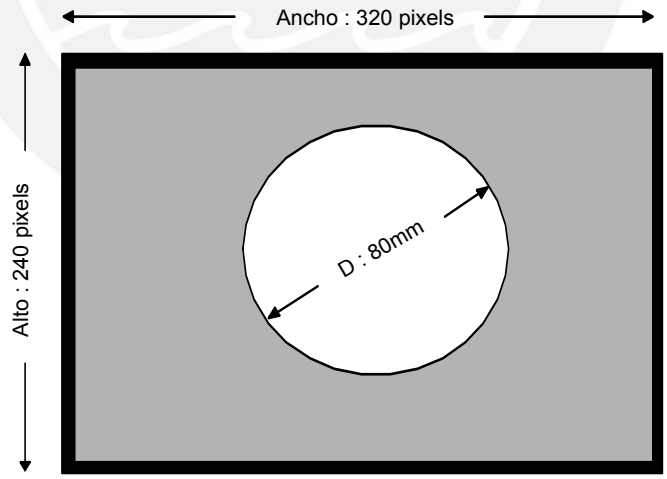



Figura 7.30 Objeto circular

d. Seleccionar el menú */configurar/calibrar,* donde aparece la pantalla que se muestra en la siguiente figura:


Figura 7.31 Presentación en mm del eje vertical y horizontal de la imagen

e. Al presionar el botón iniciar se realizara una serie de algoritmos el cual se puede resumir en lo siguiente:

- Umbralizar dejando solo el objeto de interés.
- Codificar el borde de la imagen mediante el código de cadena
- Calcular el centro del objeto para poder determinar el radio en pixeles del objeto circular. El cálculo del radio se realiza desde el centro a 4 puntos diferentes del borde del objeto, de tal manera que se halla 4 valores del radio para luego calcular un valor promedio. Entonces el resultado es el diámetro en pixeles del objeto.
- Finalmente, conociendo las dimensiones en milímetros del radio del objeto podemos calcular la dimensión horizontal (Y) y vertical (X) de la imagen de la siguiente manera:

d.....pixeles – – – – > Dmm
320pixeles – – – – > Xmm

donde **d** es la dimensión del diámetro en pixeles calculado y D es la dimensión del diámetro conocido en milímetros (80mm). En nuestro caso el resultado final es la siguiente:

```
Longitud_ejeX : 236.179mm
Longitud_ejeY : 315.000mm
```

Figura 7.32 Dimensiones en mm del eje vertical y horizontal calculado

#### 7.3.5.2 Sincronización entre la Cámara y el Robot

Para la sincronización se aplicó la siguiente técnica:

a. Ubicar aproximadamente los puntos de la imagen que se indican en la siguiente figura como punto 1 y punto 2.

b. Estos puntos se ubican con la ayuda de dos objetos circulares pequeños que sirven como guías cuyo centro se trata de ubicar en dichos puntos de la imagen (esquina superior e inferior izquierda de la imagen). Esto se realiza manualmente observando la imagen hasta que las guías se lleguen a colocar en los puntos correspondientes.

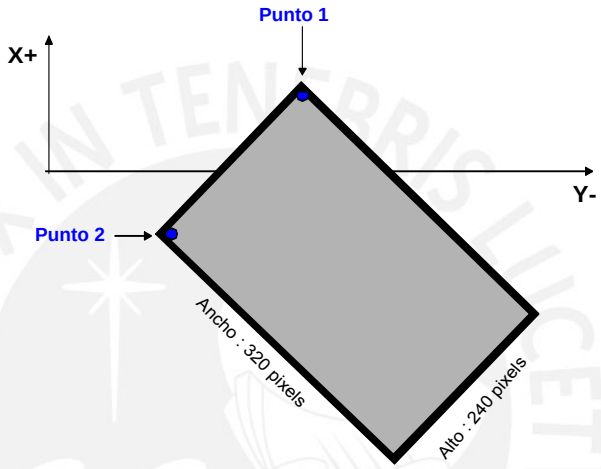



Figura 7.33 Ubicación de las guías en los puntos correspondientes

c. Luego de colocar las guías en los puntos correspondientes, con el Teach Pendant se ubica la pinza del robot con la primera guía (punto 1) una vez ubicado el TCP lo más preciso posible entonces se ingresa al menú *configura/sincronizar/* y se observara la siguiente ventana:

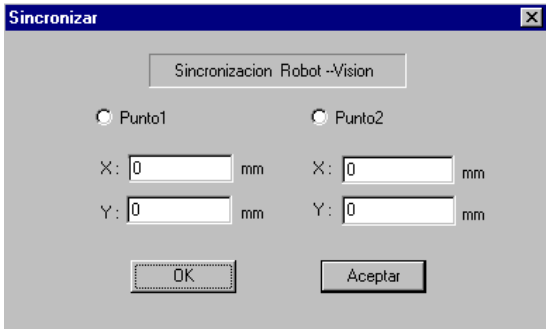



Figura 7.34 Presentación de los puntos para la sincronización

Desde aquí se hace click en punto1, y en ese instante se registra la posición X-Y del TCP que a su vez es el de la esquina superior izquierda del marco de captura.

De manera similar se hace para la segunda guía, es decir el TCP se vuelve a llevar hacia el punto 2, y una vez ubicada se registrará el punto desde la ventana mostrada anteriormente. Las coordenadas registradas en el punto 1 y punto 2 por el robot y que fueron enviadas a la PC son:

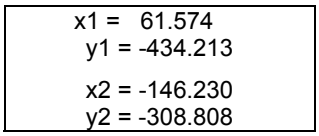



Figura 7.35 Coordenadas registradas en el punto 1 y punto 2

Parte del programa que se ejecuta en el robot y el cual se encarga de registrar las cordenadas y enviar a la PC se detalla a continuación:

```
Println " Sincronizacion Robot-Frame "
Println " ------------------------- "
for var =1 to 2
getcom 1,wait
herec pel1
println " Se Registro Cordenada X-Y "
set cordx =pvalc pel1 x
set cordy =pvalc pel1 y
set cordt=cordx/100
set var1=cordt
set var1=var1/100
set var1=var1*100
set xxl=cordt-var1
set xxh=cordt/100
sendcom 1,xxl
getcom 1,wait
sendcom 1,xxh
println "Se envio Cordenada X..."
set cordt=cordy/100
set var1=cordt
set var1=var1/100
set var1=var1*100
set yyl=cordt-var1
set yyh=cordt/100
getcom 1,wait
sendcom 1,yyl
getcom 1,wait
sendcom 1,yyh
println "Se envio Cordenada Y..."
end for
println "Finalizo Sincronizacion"
```

Figura 7.36 Programa para enviar coordenadas registradas en punto 1 y punto2

Hasta aquí se ha registrado dos puntos los cuales nos servirá para poder resolver las ecuaciones 6.1-6.6 anteriormente mencionadas en el capitulo 6:

Además, previamente a partir del código de cadena y la aplicación de los momentos se puede determinar el centroide en coordenadas de la imagen (w,h), y mediante la transformación que se haya establecido durante la fase de calibración en el Capítulo 6, se derivan las coordenadas físicas del objeto. Esa información es transmitida al ordenador que efectúa el cálculo de las trayectorias del robot.

Es importante mencionar que otro método de calibración y que también se realizó es mediante redes neuronales.

#### 7.3.5.3 Cálculo de la Orientación del Objeto.

Por otro lado, también debe sincronizarse el ángulo de rotación de la pinza con respecto al objeto, para que de esta manera el robot pueda coger el objeto de la mejor manera posible.

Para ello, primero se calcula el ángulo que debe girar la pinza de tal manera que se encuentre en posición paralela al eje w de la imagen, para luego aplicar el ángulo de orientación calculado en el Capítulo 6.

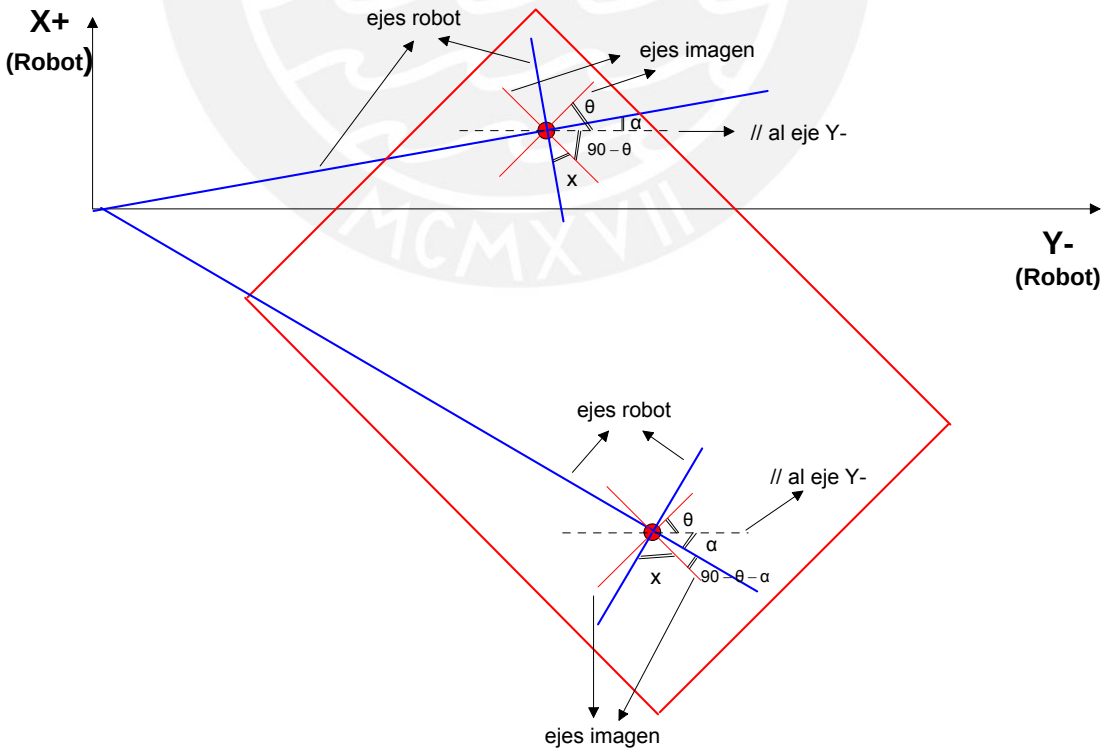


Figura 7.37 Cálculo de la Orientación del objeto

Entonces, en la figura se observa que en la parte superior se cumple:

$$x = 90 - (90 - \theta + \alpha)$$
$$x = 90 - 90 + \theta - \alpha$$
$$x = \theta - \alpha$$

Y en la parte inferior se cumple:

$$x = 90 - (90 - \theta - \alpha)$$
$$x = \theta + \alpha$$

Entonces el ángulo x, como se dijo antes, es la rotación necesaria que debe realizar la pinza, de tal manera que este se encuentre paralelo al eje horizontal de la imagen (w).

Una vez logrado este giro, entonces podemos aplicar la ecuación 6-15, el cual es la orientación respecto al eje de mínima inercia del objeto:

$$tg2\theta = 2\mu_{11} / (\mu_{20} - \mu_{02})$$

el cual despejando tenemos:

$$\theta = \frac{1}{2} atag2\mu_{11} / (\mu_{20} - \mu_{02})$$

### 7.3.6 Tarea del Robot Scorbot

Se desea utilizar el robot Scorbot ER-IX en un entorno de fabricación en el que las piezas se encuentran dispersadas en una mesa de trabajo. Las piezas escogidas para recoger deberán ser depositadas en su contenedor correspondiente como se muestra en la figura.

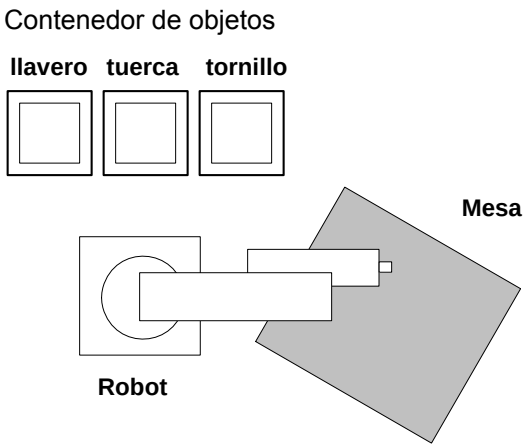


Figura 7.38 Ubicación aproximada del robot, contenedores de piezas, y mesa

El robot inicialmente estará en una posición de reposo del brazo, además deberemos tener 3 posiciones conocidas los cuales son las posiciones del contenedor.

Las posiciones deseadas para la pinza del robot en cada uno de los puntos finales (posiciones de las 3 descargas de cada pieza) se representan en el gráfico siguiente:

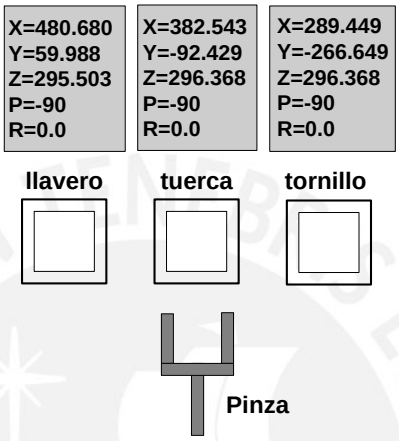



Figura 7.39 Posición final del robot en la descarga de objetos

Entonces la secuencia de operaciones realizadas por el Robot será:

a. El robot debe ir inicialmente a una posición de reposo, que tendrá como coordenadas respecto de la base:

```
X = 283.077
Y = -421.462
Z = 390.381
P = -89.016
R = 0.777
```

Figura 7.40 Coordenadas de reposo

Siempre se considerarán las posiciones y orientaciones referidas a la base del robot.

b. En esta posición el robot esperará hasta que la señal de recoger pieza se active; en ese momento se dirigirá al primer objeto a recoger y una vez ubicado correctamente se abre la pinza para luego volver a cerrarlo y coger la pieza.
c. Se esperará 1 segundo antes de hacer algún movimiento, para asegurar que la pinza se ha cerrado.

d. En función del valor de la señal enviada, se llevará la pieza a uno u otro contenedor y la depositará en ella, abriendo la pinza una vez alcanzada la posición
e. Luego, si debe recoger otro objeto deberá ir inmediatamente a la posición del objeto y realizar los mismos procedimientos anteriores.
f. Si fue el único objeto, entonces volverá a la posición de reposo a la espera de un nuevo ciclo de la tarea el cual será comunicado posteriormente por la PC.
g. Todos los movimientos se efectuarán a una velocidad de 200mm/s.
h. Se utilizarán siempre trayectorias lineales.

```
SPEEDL 150.000
SPEED   30
LABEL 1
MOVE IIII
PRINTLN "   INICIO DE MANIPUACION  "
CLRCOM 1
SETPVC TEMPI Z 385.00
GETCOM  1, WAIT
FOR VAR1= 1 TO WAIT
SENDCOM 1, 5
GETCOM  1, OBJ
IF OBJ=0
GOTO 2
ENDIF
IF OBJ=4
PRINTLN "Recogiendo Tornillo..... "
ENDIF
IF OBJ=3
PRINTLN "Recogiendo Tuerca..... "
ENDIF
IF OBJ=2
PRINTLN "Recogiendo Llavero..... "
ENDIF
GETCOM  1, POSX1
SENDCOM 1, 1
GETCOM 1, POSX2
SET POSXT=POSX2*256
SET POSXT=POSXT+POSX1
SET POSXT=POSXT*100
SETPVC TEMPI X POSXT
GETCOM  1, POSY1
SENDCOM 1, 1
GETCOM 1, POSY2
SET POSYT=POSY2*256
SET POSYT=POSYT+POSY1
SET POSYT=POSYT*100
SETPVC TEMPI Y POSYT
GETCOM  1, POSR1
SENDCOM 1, 1
GETCOM  1, POSR2
SET POSRT=POSR2*256
```

```
SET POSRT=POSRT+POSR1
SET POSRT=POSRT*100
SETP TEMPF=TEMPI
SETPVC TEMPF Z 270.000
SETPVC TEMPF Z 254.000
```

Figura 7.41 Programa de manipulación de objetos

```
SETPVC TEMPF R POSRT
MOVED TEMPI
OPEN
MOVED TEMPF
CLOSE
SETPVC TEMPF Z 320.000
MOVED TEMPF
IF OBJ=2
MOVED OBJ2
ENDIF
IF OBJ=3
MOVED OBJ3
ENDIF
IF OBJ=4
MOVED OBJ4
ENDIF
OPEN
DELAY 100
CLOSE
PRINTLN "----> PIEZA GUARDADA..."
LABEL 2
END FOR
PRINTLN ".......! Ya termine de Manipular! ......  "
GOTO 1
```

Figura 7.42 Programa de manipulación de objetos (continuación)

### 7.3.7 Interface Usuario del Sistema de Visión

En esta parte se describirá en forma sintética las funcionalidades y el modo de operación que consta la interface del operador en el reconocimiento y manipulación de objetos.

En la figura 7.43 se enumera 5 regiones, las cuales serán descritas:

1. En esta parte se visualiza la imagen de los objetos.
2. Aquí se tiene información de las características de un objeto que ha sido reconocido. Es decir se tiene información de los 4 momentos invariantes de cada objeto reconocido presionando el botón '***Siguiente'***.
3. Región donde se visualiza las coordenadas del centro de gravedad del objeto reconocido, tanto en coordenadas de la imagen (pixeles) como en las coordenadas

respecto al robot (mm). De la misma manera presionando el botón '***Siguiente'*** se tiene información de las coordenadas de cada objeto reconocido.

4. En esta parte, se selecciona la operación que deseo realizar. Es decir, por ejemplo:
   - Presionando el botón '***Capturar'***, se captura una imagen y se displaya en la región de visualización de imagen.
   - Para realizar el reconocimiento de un objeto se selecciona la opción '***Reconoce'*** y se presiona el botón ***'Continuar'.***
   - Para realizar la manipulación de objetos, una vez reconocido los objetos, se selecciona la opción ***'Manipula'*** y se presiona el botón ***'Continuar'***. Pero en este caso previamente debo seleccionar, en las casillas de selección, el orden de manipulación deseada de los objetos reconocidos,
   - En caso de que desee realizar las dos operaciones consecutivas, reconocer y manipular, se selecciona la opción '***Ciclo Total'***
5. Esta es la zona de visualización de todos los mensajes posibles a producirse durante la operación del sistema.


Figura 7.43 Interface Usuario del Sistema de Visión Artificial

# CONCLUSIONES

1. Es esta tesis se ha implementado un prototipo que incluye todas las fases clásicas de un sistema de visión artificial. Su mayor interés radica en la utilidad, versatilidad y funcionamiento, así como en la integración con el controlador del brazo robótico Scorbot ER-IX.
2. Uno de los principales problemas que hemos encontrado es la variación de la iluminación, la cual provoca sombras que dificulta la búsqueda de bordes. Otro inconveniente halllado es la geometría de la visión, es decir, la ligera variación del aspecto de los objetos cuando son vistos en diferentes posiciones.

   Se recomienda la implementación de un buen sistema de iluminación de tal manera que la imagen de los objetos se obtenga de manera homogénea durante todo el día.
3. Los algoritmos de procesamiento de imágenes utilizados han permitido tener una adecuada descripción e identificación de características de los objetos en tiempo real. Además, el empleo de los 4 momentos invariantes de Hu de menor orden, como elementos discriminadores, a sido suficiente para poder caracterizar a los objetos, el cual es posible hacerlo extensivo para caracterizar cualquier otro objeto adicional.
4. Generalmente cuando se implementa algoritmos que procesan y analizan imágenes, un problema a considerar es el tiempo de procesamiento. De este modo, las mejoras sobre los tiempos de proceso se hizo en las fases de preprocesamiento(binarización) y de extracción de las característica (código de cadena, cálculo de los momentos a partir del codigo de cadena).
5. Los procedimientos de clasificación usando redes neuronales proporcionan un buen rendimiento para el conjunto de piezas a tratar. No obstante, gracias a su diseño modular, el sistema puede ser fácilmente modificado o ampliado para incorporar nuevas características o procesos de clasificación más complejos.
6. El empleo de la redes neuronales como clasificador, implica no hacer hipótesis acerca de la distribución estadística de las variables de entrada, entonces, excepto los patrones de entrada, no ha sido necesario suministrar información adicional.

7. En lo referente al cálculo de la posición del objeto, los errores producidos son imperceptibles e irrelevantes para los requisitos de manipulación. Se ha podido realizar con éxito una tarea de pick & place de objetos previamente identificados.
8. Se ha realizado con éxito la integración del sistema de visión implementado con la celda de trabajo robotizada existente en el Laboratorio del CETAM.
9. Sí bien es cierto la aplicación desarrollada no tiene un fin específico, los resultados de este trabajo inicial en visión por computadora permitirá desarrollar aplicaciones más elaboradas como operaciones de ensamble de circuitos impresos mediante robots.

# Trabajos Futuros

El sistema de visión integrado y desarrollado es el inicio de trabajos en el cual se podría proyectar además a implementar el controlador con retroalimentación visual.

La visión artificial nos da la oportunidad de automatizar y mejorar muchos procesos industriales. En la industria se emplean cada vez más los robots, y la visión es uno de los sentidos clave para producir robots más flexibles y que puedan reaccionar ante situaciones cambiantes. Un ejemplo claro son los vehículos autónomos, que están siendo desarrollados en otros países. Por tanto el presente trabajo podría servir como base para el desarrollo de la robótica móvil.

Estudios futuros irán dirigidos a emplear otras técnicas de clasificación con otras redes neuronales de manera que se logre comparar y reducir la tasa de fallos y llegar a desarrollar un algoritmo completamente operativo.

# BIBLIOGRAFIA


1. An efficient algorithm for computation of shape moments from chain codes
   Mo Dai, Pierre Baylou Univerisité de Bordeaux-France

2. Operación automática de control de calidad utilizando un sistema de visión
   Royman López Beltrán, Edgar Sotter Solano
   Laboratorio de Robótica y Producción Automática-Universidad del Norte

3. III Jornadas Iberoamericanas de Robótica
   J.M. Ibarra CINVESTAV-Mexico

4. ESHED ROBOTEC: *Robot Vision PRO: user's manual.*
   Eshed Robotec. Israel, 1995.

5. Control, Sensing, Vision, and Intelligence.
   McGraw-Hill, New York, 1989
   K.S. Fu, R.C. González; C.S.G. Lee. *Robotics.*

6. Redes Neuronales y Reconocimiento de Patrones.
   Dr. Luis Alonso Romero, Dr. Teodoro Calonge Cano.
   Universidad de Salamanca- España - Universidad de Valladolid. España

7. Sistema de Localización y Posicionamiento de Piezas utilizando Visión Artificial.
   Javier de Lope, Francisco Serradilla, José G. Zato

8. Clasificación de patrones mediante redes neuronales artificiales
   Anales del instituto de ingenieros de chile
   Pablo estevez valencia, Abril 1999, pp 24-31

9. Selección de características para clasificadores neuronales
   Pablo Estevez valencia, Diciembre 1999, pp 65-74

10 Aprendizaje y Percepción: Preproceso y Extracción de Características
   Facultad de Informática-Universidad Politecnica de Valencia
   Enrique Vidal Ruiz, Febrero 2001

11 Reconocimiento de Patrones
   Dpto. Automática, Ingeniería Electrónica e Informática Industrial.
   Universidad Politécnica de Madrid
   UPM-DISAM

12 Del Procesamiento a la Visión Artificial
   Rafael Molina Soriano
   Dpto Ciencias de la Computacion e I.A-Universidad de Granada

# APENDICE B

## Cálculo de los Momentos Centrales a partir de los Momentos Generales

La ecuación general esta dada por:

$$u_{pq} = \sum_{r=0}^{p}\sum_{s=0}^{q}\binom{p}{r}\binom{q}{s}(-x)^r(-y)^s m_{p-r,q-s} \tag{1}$$

como podemos observar el calculo implica una sumatoria de combinatorias, para ello debemos recordar lo siguiente:

$$\binom{m}{0}=1 \qquad \binom{m}{1}=m \qquad \binom{m}{n}=\frac{|m}{|n|m-n} \tag{2}$$

### 1. Momento $U_{11}$ (p=1, q=1):

$$u_{11} = \sum_{r=0}^{1}\sum_{s=0}^{1}\binom{1}{r}\binom{1}{s}(-x)^r(-y)^s m_{1-r,1-s} \tag{3}$$

desarrollando separadamente las combinaciones:

$$r=0 \text{ y } s=0 \qquad \binom{1}{0}\binom{1}{0}(-x)^0(-y)^0 m_{1,1} = m_{11}$$

$$r=0 \text{ y } s=1 \qquad \binom{1}{0}\binom{1}{1}(-x)^0(-y)^1 m_{1,0} = -ym_{10}$$

$$r=1 \text{ y } s=0 \qquad \binom{1}{1}\binom{1}{0}(-x)^1(-y)^0 m_{0,1} = -xm_{01}$$

$$r=1 \text{ y } s=1 \qquad \binom{1}{1}\binom{1}{1}(-x)^1(-y)^1 m_{0,0} = xym_{00}$$

sumando los términos hallados:

$$u_{11} = m_{11} - ym_{10} - xm_{01} + xym_{00}$$

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

$$u_{11} = m_{11} - \frac{m_{10}}{m_{00}}m_{01} \tag{4}$$

**2. Momento $U_{20}$ (p=2, q=0):**

$$u_{20} = \sum_{r=0}^{2}\sum_{s=0}^{0}\binom{2}{r}\binom{0}{s}(-x)^r(-y)^s m_{2-r,0-s} \tag{5}$$

desarrollando separadamente las combinaciones:

$$r=0 \text{ y } s=0 \qquad \binom{2}{0}\binom{0}{0}(-x)^0(-y)^0 m_{2,0} = m_{20}$$

$$r=1 \text{ y } s=0 \qquad \binom{2}{1}\binom{0}{0}(-x)^1(-y)^0 m_{1,0} = 2(-x)m_{10}$$

$$r=2 \text{ y } s=0 \qquad \binom{2}{2}\binom{0}{0}(-x)^2(-y)^0 m_{0,0} = x^2 m_{00}$$

sumando los términos hallados:

$$u_{20} = m_{20} - 2xm_{10} + x^2 m_{00}$$

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

$$u_{20} = m_{20} - \frac{m_{10}^2}{m_{00}} \tag{6}$$

**3. Momento $U_{02}$ (p=0, q=2):**

$$u_{02} = \sum_{r=0}^{0}\sum_{s=0}^{2}\binom{0}{r}\binom{2}{s}(-x)^r(-y)^s m_{0-r,2-s} \tag{7}$$

desarrollando separadamente las combinaciones:

$$r=0 \text{ y } s=0 \qquad \binom{0}{0}\binom{2}{0}(-x)^0(-y)^0 m_{0,2} = m_{02}$$

$$r=0 \text{ y } s=1 \qquad \binom{0}{0}\binom{2}{1}(-x)^0(-y)^1 m_{0,1} = -2ym_{01}$$

$$r=0 \text{ y } s=2 \qquad \binom{0}{0}\binom{2}{2}(-x)^0(-y)^2 m_{0,0} = y^2 m_{00}$$

sumando los términos hallados:

$$u_{02} = m_{02} - 2ym_{01} + y^2 m_{00}$$

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

$$u_{02} = m_{02} - \frac{m_{01}^2}{m_{00}} \tag{8}$$

**4. Momento $U_{30}$ (p=3, q=0):**

$$u_{30} = \sum_{r=0}^{3}\sum_{s=0}^{0}\binom{3}{r}\binom{0}{s}(-x)^r(-y)^s m_{3-r,0-s} \tag{9}$$

desarrollando separadamente las combinaciones:

$$\text{r=0 y s=0} \qquad \binom{3}{0}\binom{0}{0}(-x)^0(-y)^0 m_{3,0} = m_{30}$$

$$\text{r=1 y s=0} \qquad \binom{3}{1}\binom{0}{0}(-x)^1(-y)^0 m_{2,0} = 3(-x)m_{20}$$

$$\text{r=2 y s=0} \qquad \binom{3}{2}\binom{0}{0}(-x)^2(-y)^0 m_{1,0} = 3x^2 m_{10}$$

$$\text{r=3 y s=0} \qquad \binom{3}{3}\binom{0}{0}(-x)^3(-y)^0 m_{0,0} = -x^3 m_{00}$$

sumando los términos hallados:

$$u_{30} = m_{30} - 3xm_{20} + 3x^2 m_{10} - x^3 m_{00}$$

$$u_{30} = m_{30} - 3\frac{m_{10}}{m_{00}}m_{20} + 3\frac{m_{10}^2}{m_{00}^2}m_{10} - \frac{m_{10}^3}{m_{00}^3}m_{00}$$

$$u_{30} = m_{30} - 3\frac{m_{10}}{m_{00}}m_{20} + 2\frac{m_{10}^3}{m_{00}^2} \tag{10}$$

**5. Momento $U_{03}$ (p=0, q=3):**

$$u_{03} = \sum_{r=0}^{0}\sum_{s=0}^{3}\binom{0}{r}\binom{3}{s}(-x)^r(-y)^s m_{0-r,3-s} \tag{11}$$

desarrollando separadamente las combinaciones:

$$\text{r=0 y s=0} \qquad \binom{0}{0}\binom{3}{0}(-x)^0(-y)^0 m_{0,3} = m_{03}$$

$$\text{r=0 y s=1} \qquad \binom{0}{0}\binom{3}{1}(-x)^0(-y)^1 m_{0,2} = -3ym_{02}$$

$$\text{r=0 y s=2} \qquad \binom{0}{0}\binom{3}{2}(-x)^0(-y)^2 m_{0,1} = 3y^2 m_{01}$$

$$\text{r=0 y s=3} \qquad \binom{0}{0}\binom{3}{3}(-x)^0(-y)^3 m_{0,0} = -y^3 m_{00}$$

sumando los términos hallados:

$$u_{03} = m_{03} - 3ym_{02} + 3y^2 m_{01} - y^3 m_{00}$$

$$u_{03} = m_{03} - 3\frac{m_{01}}{m_{00}}m_{02} + 3\frac{m_{01}^3}{m_{00}^2}m_{01} - \frac{m_{01}^3}{m_{00}^2}$$

$$u_{03} = m_{03} - 3\frac{m_{01}}{m_{00}}m_{02} + 2\frac{{m_{01}}^3}{m_{00}^2} \tag{12}$$

## 6. Momento $U_{21}$ (p=2, q=1):

$$u_{21} = \sum_{r=0}^{2}\sum_{s=0}^{1}\binom{2}{r}\binom{1}{s}(-x)^r(-y)^s m_{2-r,1-s} \tag{13}$$

desarrollando separadamente las combinaciones:

$$r=0 \text{ y } s=0 \qquad \binom{2}{0}\binom{1}{0}(-x)^0(-y)^0 m_{2,1} = m_{21}$$

$$r=0 \text{ y } s=1 \qquad \binom{2}{0}\binom{1}{1}(-x)^0(-y)^1 m_{2,0} = -ym_{20}$$

$$r=1 \text{ y } s=0 \qquad \binom{2}{1}\binom{1}{0}(-x)^1(-y)^0 m_{1,1} = 2(-x)m_{11}$$

$$r=1 \text{ y } s=1 \qquad \binom{2}{1}\binom{1}{1}(-x)^1(-y)^1 m_{1,0} = 2(-x)(-y)m_{10}$$

$$r=2 \text{ y } s=0 \qquad \binom{2}{2}\binom{1}{0}(-x)^2(-y)^0 m_{0,1} = x^2 m_{01}$$

$$r=2 \text{ y } s=1 \qquad \binom{2}{2}\binom{1}{1}(-x)^2(-y)^1 m_{0,0} = x^2(-y)m_{00}$$

sumando los términos hallados:

$$u_{21} = m_{21} - ym_{20} - 2xm_{11} + 2xym_{10} + x^2 m_{01} - x^2 ym_{00}$$

$$u_{21} = m_{21} - \frac{m_{01}}{m_{00}}m_{20} - 2\frac{m_{10}}{m_{00}}m_{11} + 2\frac{m_{10}m_{01}}{m_{00}^2}m_{10} + \frac{{m_{10}}^2}{m_{00}^2}m_{01} - \frac{{m_{10}}^2 m_{01}}{m_{00}^3}m_{00}$$

$$u_{21} = m_{21} - \frac{m_{01}}{m_{00}}m_{20} - 2\frac{m_{10}}{m_{00}}m_{11} + 3\frac{{m_{10}}^2 m_{01}}{m_{00}^2} - \frac{{m_{10}}^2}{m_{00}^2}m_{01}$$

$$u_{21} = m_{21} - \frac{m_{01}}{m_{00}}m_{20} - 2\frac{m_{10}}{m_{00}}m_{11} + 2\frac{{m_{10}}^2 m_{01}}{m_{00}^2} \tag{14}$$

## 7. Momento $U_{12}$ (p=1, q=2):

$$u_{12} = \sum_{r=0}^{1}\sum_{s=0}^{2}\binom{1}{r}\binom{2}{s}(-x)^r(-y)^s m_{1-r,2-s} \tag{15}$$

desarrollando separadamente las combinaciones:

r=0 y s=0 $\quad \binom{1}{0}\binom{2}{0}(-x)^0(-y)^0 m_{1,2} = m_{12}$

r=0 y s=1 $\quad \binom{1}{0}\binom{2}{1}(-x)^0(-y)^1 m_{1,1} = 2(-y)m_{11}$

r=0 y s=2 $\quad \binom{1}{0}\binom{2}{2}(-x)^0(-y)^2 m_{1,0} = y^2 m_{10}$

r=1 y s=0 $\quad \binom{1}{1}\binom{2}{0}(-x)^1(-y)^0 m_{0,2} = -xm_{02}$

r=1 y s=1 $\quad \binom{1}{1}\binom{2}{1}(-x)^1(-y)^1 m_{0,1} = 2xym_{01}$

r=1 y s=2 $\quad \binom{1}{1}\binom{2}{2}(-x)^1(-y)^2 m_{0,0} = -xy^2 m_{00}$

sumando los términos hallados:

$$u_{12} = m_{12} - 2\frac{m_{01}}{m_{00}}m_{11} + \frac{m_{01}}{m_{00}^{\,2}}m_{10} - \frac{m_{10}}{m_{00}}m_{02} + 2\frac{m_{01}^{\,2}}{m_{00}^{\,2}}m_{10} - \frac{m_{01}^{\,2}}{m_{00}^{\,2}}m_{10}$$

$$u_{12} = m_{12} - 2\frac{m_{01}}{m_{00}}m_{11} - \frac{m_{10}}{m_{00}}m_{02} + 2\frac{m_{01}^{\,2}}{m_{00}^{\,2}}m_{10} \tag{16}$$

# APENDICE C

## Cálculo de los momentos generales a partir del código de cadena

1. Primero mencionemos algunas de las relaciones de igualdad empleadas:

$$y_c^4 - y_{c-1}^4 = (y_c^2 - y_{c-1}^2)(y_c^2 + y_{c-1}^2)$$
$$y_c^4 - y_{c-1}^4 = (y_c - y_{c-1})(y_c + y_{c-1})(y_c^2 + y_{c-1}^2)$$
$$y_c^4 - y_{c-1}^4 = dy_c(y_c + y_c - dy_c)(y_c^2 + y_c^2 + dy_c^2 - 2y_c dy_c)$$
$$y_c^4 - y_{c-1}^4 = dy_c(2y_c - dy_c)(2y_c^2 + dy_c^2 - 2y_c dy_c)$$
$$\frac{y_c^4 - y_{c-1}^4}{4} = y_c(\frac{2y_c - dy_c}{2})(\frac{2y_c^2 + dy_c^2 - 2y_c y_c}{2})$$
$$\frac{y_c^4 - y_{c-1}^4}{4} = dy_c(y_c - \frac{dy_c}{2})(y_c^2 + \frac{dy_c^2}{2} - y_c dy_c) \quad (1)$$

$$y_c^3 - y_{c-1}^3 = (y_c - y_{c-1})(y_c^2 + y_c y_{c-1} + y_{c-1}^2)$$
$$y_c^3 - y_{c-1}^3 = dy_c\left[y_c^2 + y_c(y_c - dy_c) + y_c^2 + dy_c^2 - 2y_c dy_c\right]$$
$$y_c^3 - y_{c-1}^3 = dy_c\left[y_c^2 + y_c^2 - y_c dy_c + y_c^2 + dy_c^2 - 2y_c dy_c\right]$$
$$y_c^3 - y_{c-1}^3 = dy_c(3y_c^2 - 3y_c dy_c + dy_c^2)$$
$$\frac{y_c^3 - y_{c-1}^3}{3} = dy_c(\frac{3y_c^2 - 3y_c dy_c + dy_c^2}{3})$$
$$\frac{y_c^3 - y_{c-1}^3}{3} = dy_c(y_c^2 - y_c dy_c + \frac{dy_c^2}{3}) \quad (2)$$

2. Cálculo de $m_{10}$

$$m_{10} = \frac{1}{1+0+2}\int_C x^{1+1}y^0dy - x^1y^{0+1}dx$$

$$m_{10} = \frac{1}{3}\int_C x^2dy - xydx$$

$$m_{10} = \frac{1}{3}\int_C x^2mdx - x(mx - mx_c + y_c)dx$$

$$m_{10} = \frac{1}{3}\int_C (x^2m - x^2m + xmx_c - xy_c)dx$$

$$m_{10} = \frac{1}{3}\int_C (mx_c - y_c)xdx$$

$$m_{10} = \frac{1}{3}\sum_{c=1}^{N}(mx_c - y_c)\left[\frac{x^2}{2}\Big|_{xc-1}^{xc}\right]$$

$$m_{10} = \frac{1}{3}\sum_{c=1}^{N}(mx_c - y_c)(\frac{x_c^2 - x_{c-1}^2}{2})$$

$$m_{10} = \frac{1}{3}\sum_{c=1}^{N}(mx_c - y_c)(\frac{2x_c - dx_c}{2})dx_c$$

$$m_{10} = \frac{1}{3}\sum_{c=1}^{N}(mx_c - y_c)(\frac{2x_c - dx_c}{2})dx_c$$

$$m_{10} = \frac{1}{3}\sum_{c=1}^{N}(x_cdy_c - y_cdx_c)(x_c - \frac{dx_c}{2})$$

$$m_{10} = \frac{1}{3}\sum_{c=1}^{N}Ac(x_c - \frac{dx_c}{2}) \qquad (3)$$

3. Cálculo de $m_{01}$

$$m_{01} = \frac{1}{0+1+2}\int_C x^{0+1}y^1dy - x^0y^{1+1}dx$$

$$m_{01} = \frac{1}{3}\int_C xydy - y^2dx$$

$$m_{01} = \frac{1}{3}\int_C (\frac{y - y_c}{m} + x_c)ydy - \frac{y^2}{m}dy$$

$$m_{01} = \frac{1}{3} * \frac{1}{m}\int_C (y^2 - yy_c + mx_cy - y^2)dy$$

$$m_{01} = \frac{1}{3} * \frac{1}{m}\int_C (mx_c - y_c)ydy$$

$$m_{01} = \frac{1}{3}\sum_{c=1}^{N}(mx_c - y_c)\left[\frac{y^2}{2}\Big|_{yc-1}^{yc}\right]$$

$$m_{01} = \frac{1}{3}\sum_{c=1}^{N}(mx_c - y_c)(\frac{y_c^2 - y_{c-1}^2}{2})$$

$$m_{01} = \frac{1}{3}\sum_{c=1}^{N}(mx_c - y_c)(\frac{2y_c - dy_c}{2})dy_c$$

$$m_{01} = \frac{1}{3}\sum_{c=1}^{N}(x_c dy_c - y_c dx_c)(y_c - \frac{dy_c}{2})$$

$$m_{10} = \frac{1}{3}\sum_{c=1}^{N}Ac(y_c - \frac{dy_c}{2}) \quad (4)$$

4. Cálculo de $m_{20}$

$$m_{20} = \frac{1}{2+0+2}\int_C x^{2+1}y^0 dy - x^2 y^{0+1}dx$$

$$m_{20} = \frac{1}{4}\int_C x^3 dy - x^2 y dx$$

$$m_{20} = \frac{1}{4}\int_C x^3 m dx - x^2(mx - mx_c + y_c)dx$$

$$m_{20} = \frac{1}{4}\int_C (x^3 m - x^3 m + x^2 m x_c - x^2 y_c)dx$$

$$m_{20} = \frac{1}{4} * \frac{1}{m}\int_C (mx_c - y_c)x^2 dx$$

$$m_{20} = \frac{1}{4}\sum_{c=1}^{N}(mx_c - y_c)\left[\frac{x^3}{3}\Big|_{xc-1}^{xc}\right]$$

$$m_{20} = \frac{1}{4}\sum_{c=1}^{N}(mx_c - y_c)dx_c(x_c^2 - x_c dx_c + \frac{dx_c^2}{3})$$

$$m_{20} = \frac{1}{4}\sum_{c=1}^{N}(mx_c - y_c)dx_c(x_c^2 - x_c dx_c + \frac{dx_c^2}{3})$$

$$m_{20} = \frac{1}{4}\sum_{c=1}^{N}(x_c dy_c - y_c dx_c)(x_c^2 - x_c dx_c + \frac{dx_c^2}{3})$$

$$m_{20} = \frac{1}{4}\sum_{c=1}^{N}A_c\left[x_c^2 - x_c dx_c + \frac{1}{3}dx_c^2)\right] \quad (5)$$

4. Cálculo de $m_{02}$

$$m_{02} = \frac{1}{0+2+2}\int_C x^{0+1}y^2 dy - x^0 y^{2+1}dx$$

$$m_{02} = \frac{1}{4}\int_C xy^2 dy - y^2 dx$$

$$m_{02} = \frac{1}{4}\int_C y^2(\frac{y - y_c + mx_c}{m})dy - \frac{y^3}{m}dy$$

$$m_{02} = \frac{1}{4} * \frac{1}{m} \int_C (mx_c - y_c) y^2 dy$$

$$m_{02} = \frac{1}{4} * \frac{1}{m} \sum_{c=1}^{N} (mx_c - y_c) \left[ \frac{y^3}{3} \Big|_{yc-1}^{yc} \right]$$

$$m_{02} = \frac{1}{4} * \frac{1}{m} \sum_{c=1}^{N} (mx_c - y_c) dy_c \left( y_c^2 - y_c dy_c + \frac{dy_c^2}{3} \right)$$

$$m_{02} = \frac{1}{4} \sum_{c=1}^{N} (x_c dy_c - y_c dx_c) \left( y_c^2 - y_c dy_c + \frac{dy_c^2}{3} \right)$$

$$m_{02} = \frac{1}{4} \sum_{c=1}^{N} A_c \left[ y_c^2 - y_c dy_c + \frac{dy_c^2}{3} \right] \quad (6)$$

5. Cálculo de $m_{11}$

$$m_{11} = \frac{1}{1+1+2} \int_C x^{1+1} y^1 dy - x^1 y^{1+1} dx$$

$$m_{11} = \frac{1}{4} \int_C x^2 y dy - x y^2 dx$$

$$m_{11} = \frac{1}{4} \int_C x^2 (mx - mx_c + y_c) m dx - x (mx - mx_c + y_c)^2 dx$$

$$m_{11} = \frac{1}{4} \int_C (m^2 x^3 - m^2 x_c x^2 + m y_c x^2 - (m^2 x^2 + m^2 x_c^2 - 2m^2 x x_c + y_c^2 + 2y_c mx - 2y_c mx_c) x dx$$

$$m_{11} = \frac{1}{4} \int_C (m^2 x^3 - m^2 x_c x^2 + m y_c x^2 - m^2 x^3 - m^2 x x_c^2 + 2m^2 x^2 x_c - x y_c^2 - 2y_c mx^2 + 2y_c mx x_c) dx$$

$$m_{11} = \frac{1}{4} \int_C (m^2 x_c x^2 - m y_c x^2 - m^2 x x_c^2 - x y_c^2 + 2y_c mx x_c) dx$$

$$m_{11} = \frac{1}{4} \int_C (x^2 m (mx_c - y_c) - x (mx_c - y_c)^2) dx$$

$$m_{11} = \frac{mx_c - y_c}{4} \sum_{c=1}^{N} \left[ m \frac{x^3}{3} \Big|_{xc-1}^{xc} - (mx_c - y_c) \frac{x^2}{2} \Big|_{xc-1}^{xc} \right]$$

$$m_{11} = \frac{mx_c - y_c}{4} \sum_{c=1}^{N} \left[ m dx_c \left( x_c^2 - x_c dx_c + \frac{dx_c^2}{3} \right) - \frac{dx}{2} (mx_c - y_c)(2x_c - dx_c) \right]$$

$$m_{11} = \sum_{c=1}^{N} \frac{x_c dy_c - y_c dx_c}{4} \left[ \frac{2mx_c^2 - 2mx_c dx_c + \frac{2m dx_c^2}{3} - 2mx_c^2 + 2x_c y_c + mx_c dx_c - y_c dx_c)}{2} \right]$$

$$m_{11} = \sum_{c=1}^{N} \frac{x_c dy_c - y_c dx_c}{4} \left[ \frac{-mx_c dx_c + \frac{2dx_c dy_c}{3} + 2x_c y_c - y_c dx_c)}{2} \right]$$

$$m_{11} = \sum_{c=1}^{N} \frac{x_c dy_c - y_c dx_c}{4} \left[ \frac{2x_c y_c + \frac{2dx_c dy_c}{3} - (x_c dy_c + y_c dx_c)}{2} \right]$$

$$m_{11} = \frac{1}{4}\sum_{c=1}^{N} A_c\left[x_c y_c - \frac{1}{2}(x_c dy_c + y_c dx_c) + \frac{1}{3}dx_c dy_c)\right] \quad (7)$$

6. Cálculo de $m_{30}$

$$m_{30} = \frac{1}{3+0+2}\int_C x^{3+1}y^0 dy - x^3 y^{0+1}dx$$

$$m_{30} = \frac{1}{5}\int_C x^4 dy - x^3 y dx$$

$$m_{30} = \frac{1}{5}\int_C x^4 m dx - x^3(yc + mx - mx_c)dx$$

$$m_{30} = \frac{1}{5}\int_C x^4 m dx - x^3 y_c dx - x^4 m dx + x^3 m x_c dx$$

$$m_{30} = \frac{1}{5}\int_C (x^3 m x_c - x^3 yc)dx$$

$$m_{30} = \frac{1}{5}(mx_c - y_c)\int_C x^3 dx$$

$$m_{30} = \frac{1}{5}\sum (mx_c - y_c)\left[\frac{x^4}{4}\Big|_{xc-1}^{xc}\right]$$

$$m_{30} = \frac{1}{5} * \frac{1}{4}\sum (mx_c - y_c)(x_c^4 - x_{c-1}^4)$$

$$m_{30} = \frac{1}{5}\sum (mx_c - y_c)(dx_c)(x_c - \frac{dx_c}{2})(x_c^2 + \frac{dx_c^2}{2} - x_c dx_c)$$

$$m_{30} = \frac{1}{5}\sum (x_c dy_c - y_c dx_c)(x_c - \frac{dx_c}{2})(x_c^2 + \frac{dx_c^2}{2} - x_c dx_c)$$

$$m_{30} = \frac{1}{5}\sum Acc(x_c - \frac{dx_c}{2})(x_c^2 + \frac{dx_c^2}{2} - x_c dx_c) \quad (8)$$

7. Cálculo de $m_{03}$

$$m_{03} = \frac{1}{0+3+2}\int_C x^1 y^3 dy - x^0 y^4 dx$$

$$m_{03} = \frac{1}{5}\int_C x^1 y^3 dy - y^4 dx$$

$$m_{03} = \frac{1}{5}\int_C \frac{(y - yc + mx_c)}{m} y^3 dy - \frac{y^4}{m}dy$$

$$m_{03} = \frac{1}{5m}\int_C (y^4 - y^3 y_c + mx_c y^3)dy - y^4 dy$$

$$m_{03} = \frac{1}{5m}\int_C (y^3 m x_c - y^3 yc)dy$$

$$m_{03} = \frac{1}{5m}(mx_c - y_c)\int_C y^3 dy$$

$$m_{03} = \frac{1}{5m}\sum (mx_c - y_c)\left[\frac{y^4}{4}\Big|_{yc-1}^{yc}\right]$$

$$m_{30} = \frac{1}{5m} * \frac{1}{4}\sum (mx_c - y_c)(y_c^4 - y_{c-1}^4)$$

$$m_{30} = \frac{1}{5m}\sum (mx_c - y_c)(dy_c)(y_c - \frac{dy_c}{2})(y_c^2 + \frac{dy_c^2}{2} - y_c dy_c)$$

$$m_{30} = \frac{1}{5}\sum (x_c dy_c - y_c dx_c)(y_c - \frac{dy_c}{2})(y_c^2 + \frac{dy_c^2}{2} - y_c dy_c)$$

$$m_{30} = \frac{1}{5}\sum Acc(y_c - \frac{dy_c}{2})(y_c^2 + \frac{dy_c^2}{2} - y_c dy_c) \quad (9)$$

8. Cálculo de $m_{21}$ y $m_{12}$

Puesto que los momentos de orden $m_{21}$ y $m_{12}$ presentan puntos de indeterminación, estos se evaluaran de otra manera:

$$m_{pq} = \sum_{i=1}^{n} D_i$$

donde p=1 y q=2 o p=2 y q=1;

Contribución de cada segmento cuando la pendiente es infinito:

$$Di = \int_{ci} \frac{x^{p+1}}{p+1} y^q dy = \int_{yi} \frac{x_i^{p+1}}{p+1} dy$$

$$Di == \int_{xi}^{xi+1} \frac{x^{p+1}}{p+1} y^q dy = \int_{yi}^{yi+1} \frac{x_i^{p+1}}{p+1} y^q dy = x_i^{p+1} \frac{y_{i+1}^{q+1} - y_i^{q+1}}{(p+1)(q+1)}$$

$$m_{pq} = \sum_{i=1}^{N} x_i^{p+1} \frac{y_{i+1}^{q+1} - y_{i+1}^{q+1}}{(p+1)(q+1)} \quad (10)$$

# Pruebas y Resultados

Ya en la etapa de desarrollo del software hemos dado algunas ideas de funcionamiento del sistema implementado, sin embargo a modo de ilustrar el funcionamiento completo y las pruebas del sistema mostraremos en forma secuencial los resultados mediante algunas gráficas.

## 1. Calibrando el Cuadro de la imagen de Captura

Antes de todo, lo primero que debe realizarse es calcular las dimensiones en milímetros del cuadro de captura de la imagen. Esto es necesario ya que la distancia de la cámara a la mesa puede haber sido alterado por algún motivo.

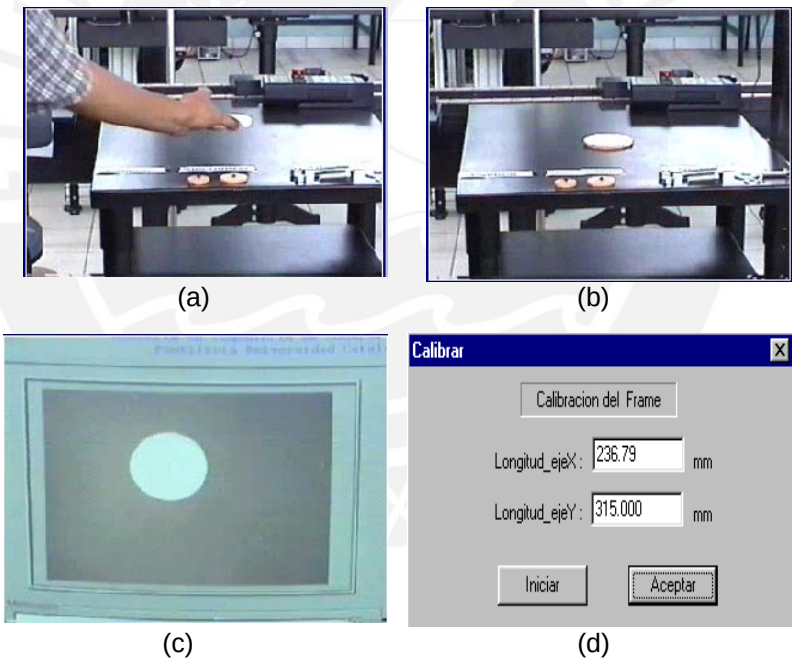



Figura 1. Calibrando el cuadro de captura

En la figura 1a. se toma el objeto circular de diámetro conocido y se coloca en la mesa de trabajo como se observa en la figura 1b. Luego observamos la imagen del objeto circular en el software de visión para obtener las correspondientes dimensiones en milímetros del cuadro de captura de la imagen, como se muestra en la figura 1d.

## 2. Sincronizando las Coordenadas de la Imagen y las Coordenadas del espacio del Robot.

En este caso debemos colocar dos guías (figura 2a y 2c) como se menciono anteriormente. La parte central de las guías deben ser colocadas en la parte superior e inferior izquierda del cuadro de imagen (figura 2e y 2f).

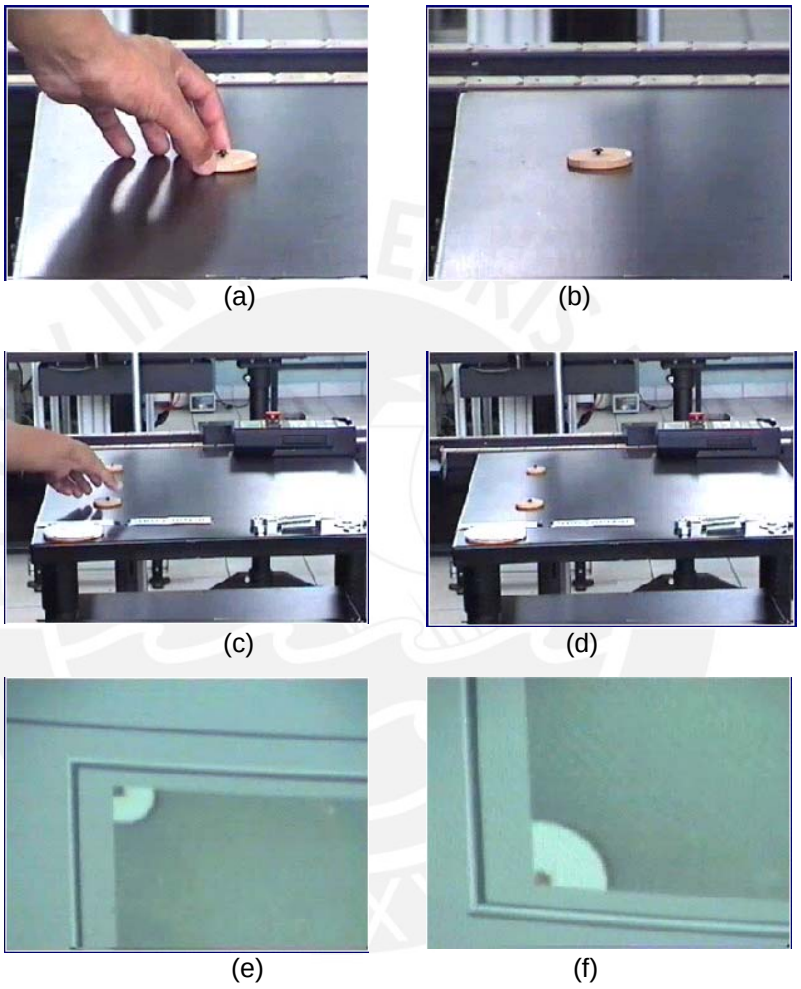
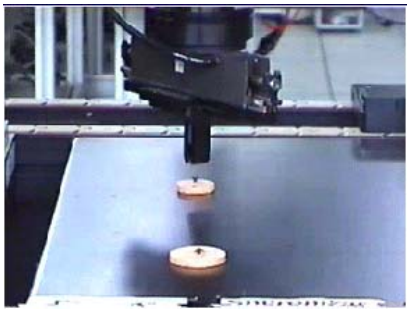
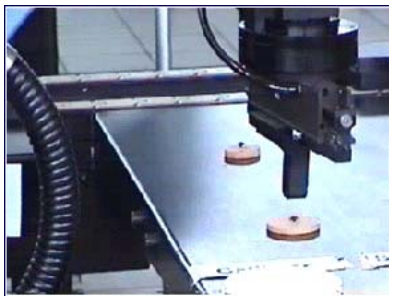



Figura 2. Sincronizando Robot - Visión

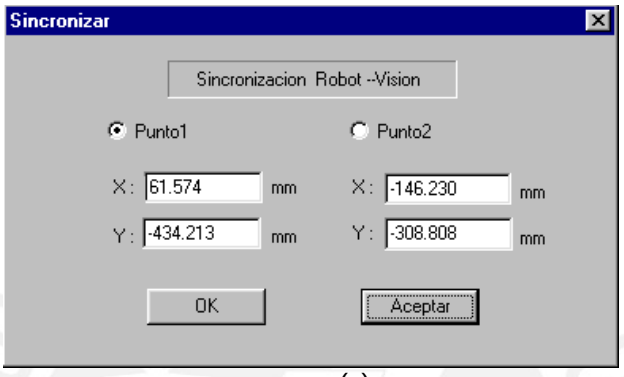
Luego llevamos el centro de la pinza del robot (TCP) en ambas posiciones como se ve en la figura 3a y 3b y así obtenemos las coordenadas mostradas en la figura 3c. Estas coordenadas nos servirán para calcular las transformación de escala desde la coordenada imagen (pixeles) a coordenadas de movimiento del robot (milímetros)



(a)



(b)

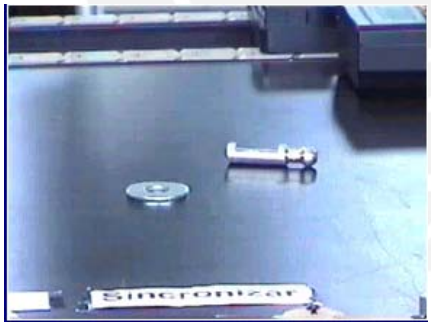



(c)

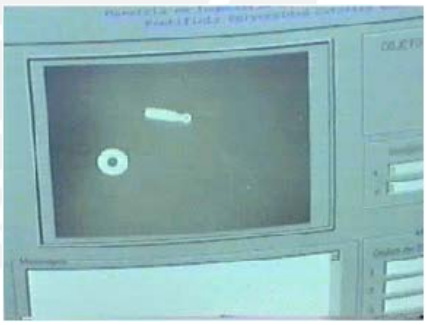
Figura 3. Sincronizando Robot – Visión (continuación)

## 3. Reconocimiento de Objetos

Para ver el funcionamiento de reconocimiento colocamos dos objetos, una arandela y un llavero (figura 4a y 4b) luego vemos los resultados en 4d.

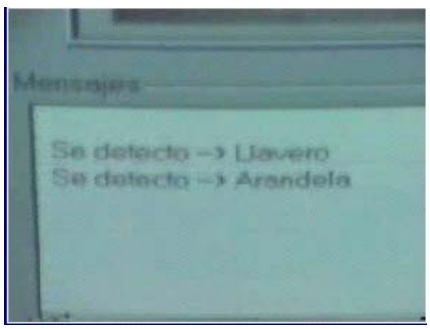


(a)



(b)


(c)




(d)

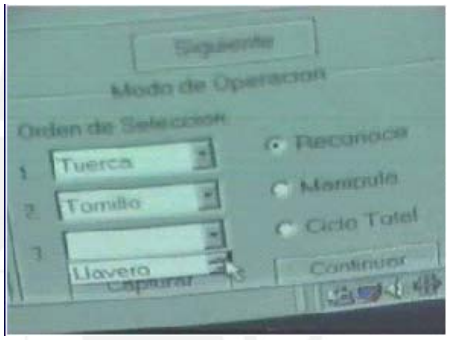
Figura 4. Prueba de Reconocimiento de Objetos

## 4. Manipulación de Objetos

Finalmente probado el reconocimiento, se puede pasar a la manipulación. En esta prueba colocamos 3 objetos, tornillo, llavero y tuerca (figura 5a). Luego seleccionamos el orden de manipulación de los objetos (figura 5b). El orden de manipulación seleccionada en este caso es: primero la tuerca, luego el tornillo y finalmente el llavero.

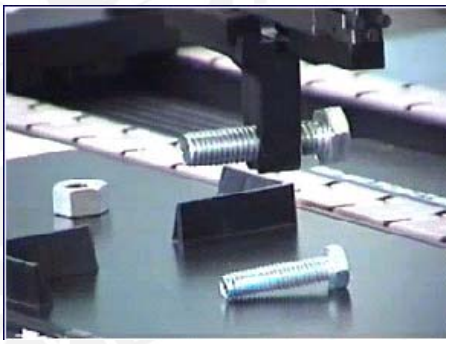
En las figuras 5c, 5d y 5f se muestra al robot manipulando la tuerca, el tornillo y finalmente el llavero.

(a)

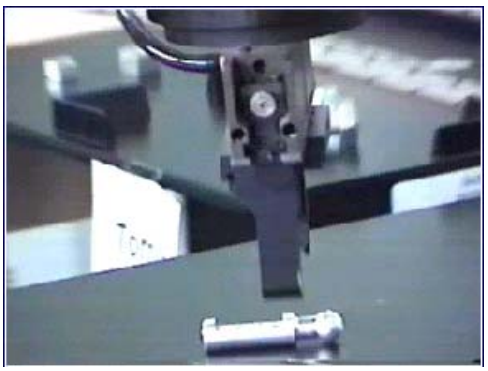



(b)

(c)



(d)



(e)

Figura 5. Prueba de Manipulación de Objetos