SIBACORE: SISTEMA BASADO EN CONOCIMIENTOS PARA EL RECONOCIMIENTO DE ELEMENTOS MECÁNICOS

JAIRO CAÑÓN RODRÍGUEZ
Departamento de Ingeniería Mecánica.
e-mail: jcanon@perseus.unalmed.edu.co

DEMETRIO ARTURO OVALLE CARRANZA
Departamento de Sistemas y Administración.
e-mail: dovalle@perseus.unalmed.edu.co
Facultad de Minas, Universidad Nacional de Colombia - Sede Medellín.

RESUMEN

El objetivo del artículo es presentar inicialmente los fundamentos de los Sistemas Avanzados de Visión utilizados en el procesamiento y reconocimiento de imágenes. Luego, propone la arquitectura del modelo del sistema de visión basado en conocimientos SIBACORE, para el reconocimiento de elementos mecánicos. Se analizan en detalle los algoritmos que son utilizados en cada una de las etapas de reconocimiento y diagnóstico, específicamente de los elementos mecánicos: arandelas y tornillos. Finalmente, se tratan los aspectos de la implementación computarizada del prototipo del sistema SIBACORE (Sistema Basado en Conocimientos para el Reconocimiento de elementos Mecánicos) el cual trabaja sobre contornos de piezas obtenidos por preprocesamiento.

PALABRAS CLAVES
Visión Artificial, Sistemas Basados en Conocimientos, Diagnóstico de Fallas, Reconocimiento de Patrones, Procesamiento de Imágenes.

ABSTRACT

The paper firstly aims at describing the foundations of Advanced Computer Vision Systems mostly used in image processing and recognition. Then, it proposes the structure of the model for SIBACORE, knowledge based vision system for mechanical image recognition. It is discussed in detail algorithms which are used in each of steps for recognition and diagnosis specifically for candle sockets and screws. Finally, computer implementation issues are described for the SIBACORE (Knowledge Based System for Mechanical Element Recognition) system prototype which works on contours of pieces obtained by preprocessing.

KEY WORDS
Computer Vision, Knowledge Based Systems, Fault System Diagnosis, Pattern Recognition, Image Processing.

1. INTRODUCCIÓN

El artículo empieza haciendo una descripción de los procesos de la Visión Artificial, y a continuación explica algunas estructuras de Sistemas de Visión de Propósito General (SVPG), para dar así un resumen del estado del arte. Luego se pasa a definir la estructura del modelo SIBACORE, con el cual se identifican elementos mecánicos de imágenes capturadas mediante una cámara de video, este sistema se desarrollo como parte de la Investigación "Diseño y Desarrollo Metodológico de Sistemas Basados en Conocimientos Distribuidos y Cooperantes Aplicados a la Ingeniería" financiada por Colciencias. El articulo finaliza comentando la manera como se implementó

el prototipo del modelo del sistema basado en conocimientos SIBACORE, destinado al reconocimiento y diagnóstico de elementos mecánicos para lo cual se describe la forma en que se realizó el preprocesado de las imágenes por medio de aplicaciones existentes en el mercado, hasta obtener imágenes binarias de los contornos; la realización de algoritmos para el procesamiento intermedio en la detección de círculos por medio de la transformada de Hough y encontrar líneas rectas por la aplicación de filtros. Como estos algoritmos se fundamentan en la representación de líneas rectas y círculos en la pantalla gráfica del computador, también se ha incluido una explicación de los Algoritmos de Bresenham para la representación de líneas rectas y círculos. La aplicación de reconocimiento de Elementos Mecánicos se hace sobre arandelas, tornillos y engranajes. En el artículo se presentan imágenes procesadas de Elementos Mecánicos, en las cuales predominan los círculos y líneas rectas como parte de los contornos.

2. SISTEMAS DE VISIÓN ARTIFICIAL

La *Visión Artificial* también llamada *Visión por Computador*, se ha definido como los procesos de obtención, caracterización e interpretación de información de imágenes tomadas de un mundo tridimensional. **[4]**. Estos procesos pueden a su vez ser divididos en seis áreas:

1. **Adquisición o captación:** Proceso a través del cual se obtiene la imagen visual.
2. **Preprocesamiento:** Proceso que incluye técnicas de reducción del ruido y realce de detalles.
3. **Segmentación:** Proceso que divide una imagen en objetos que son de nuestro interés.
4. **Descripción:** Proceso por el cual se obtienen características convenientes para diferenciar un objeto de otro.
5. **Reconocimiento:** Es el proceso que identifica cada uno de los objetos en la imagen.
6. **Interpretación**: Proceso que le asocia un significado a un conjunto de objetos reconocidos.

Las divisiones tomadas por otros autores sobre estas áreas de importancia en la visión artificial, son de forma. Para ilustrar lo anterior incluimos la figura 1, que muestra las etapas de la visión artificial según Maravall. **[5].**

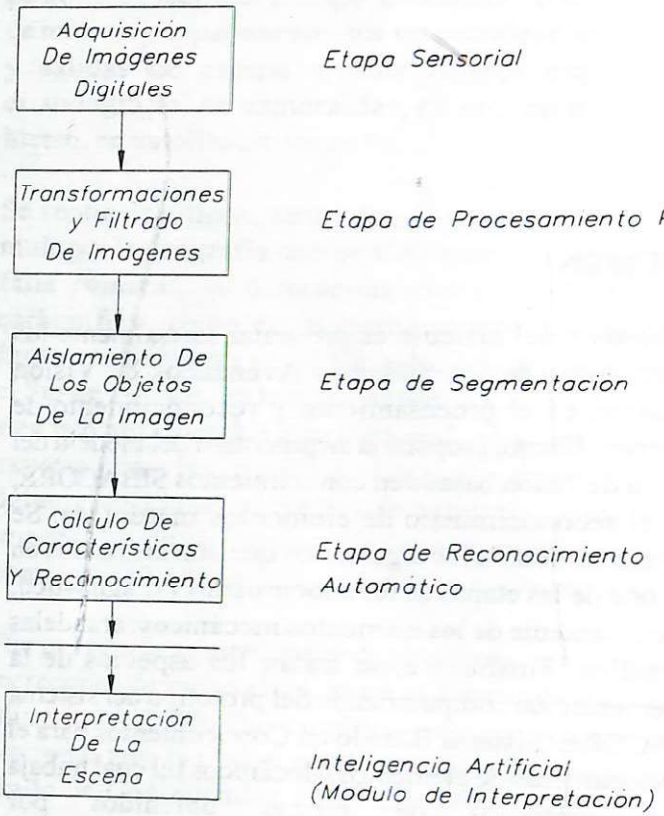


Figura 1. Etapas en la que puede organizarse un Sistema de Visión Artificial

Las áreas antes descritas se hacen necesario agruparlas de acuerdo a las complicaciones y delicadezas que llevan consigo en la implementación. Por lo tanto se pueden considerar tres niveles de procesamiento **[3], [4]:**

1. **Procesamiento de Bajo Nivel:** Asociado con los procesos primarios que pueden ser considerados como reacciones automáticas sin requerir ningún tipo de inteligencia. Es la adquisición y preprocesamiento de imágenes.

2. **Procesamiento de Nivel Intermedio:** Son los procesos que caracterizan y etiquetan componentes de la imagen que se obtiene de la visión a bajo nivel. Aquí se trata de la segmentación, la descripción y el reconocimiento de objetos individuales.

3. **Procesamiento de Alto Nivel:** Son los procesos que emulan la cognición. Los dos niveles anteriores llevan a actividades bien definidas mientras que el conocimiento y compresión de los procesos de visión de alto nivel son mas difusos y especulativos.

Estos procesos y divisiones en áreas están basadas en la forma en que se implementán los sistemas de visión artificial y no representan modelos de la visión humana.

3. SISTEMAS DE VISIÓN ARTIFICIAL BASADOS EN CONOCIMIENTOS

Los Sistemas Basados en Conocimientos para el procesamiento de imágenes fueron creados para afrontar algunas dificultades en la visión artificial como son [6]:

- La validación de la calidad de la imagen
- La selección de operadores de procesamiento apropiados y de parámetros óptimos de ejecución.
- La selección de una planificación eficaz.
- La necesidad de ajustar cuidadosamente los descriptores de la imagen y los operadores por medio de experiencias de ensayo y error.
- La falta total de herramientas objetivas de evaluación de los resultados obtenidos.

Estos sistemas requieren de un "conocimiento experto" el cual comprende diversos tipos de conocimiento sobre los operadores, su aplicabilidad, su uso y sobre modalidades de secuenciamiento de estos. También requiere de un conocimiento sobre la escena y el campo de aplicación. Este experto debe ser capaz de juzgar resultados y modelar o corregir la estrategia de análisis en curso de ejecución.

Existen dos tareas que pueden ser identificadas en los sistemas de visión artificial: la tarea de segmentación y la de interpretación. La tarea de segmentación consiste en delimitar e identificar las zonas de interés, o sea aquellas zonas que serán utilizadas por la tarea de interpretación. La segmentación tiene como objetivo encontrar la correspondencia espacial entre primitivas de la imagen y los componentes de la escena. La interpretación tiene por objeto la búsqueda de una correspondencia semántica entre primitivas y etiquetas simbólicas. La búsqueda de la mejor correspondencia es el principal reto para ambas tareas. Una de las dificultades encontradas en visión artificial concierne al paradigma del razonamiento, las cuales se pueden resumir en:

- Dificultad de modelar el razonamiento experto de análisis en oposición a la búsqueda de una secuencia óptima de procesos.
- Inhabilidad de los sistemas de visión artificial para juzgar los resultados y así poder evaluar el desempeño de un operador en la aplicación o de una secuencia de procesos sobre la imagen.
- La dificultad de corregir y reiniciar de la mejor manera la estrategia de análisis en curso de ejecución.

El propósito de los sistemas de visión artificial es ofrecer al usuario facilidades de planificación y generación automática de secuencias óptimas de procesamientos, facilidades de descripción de los problemas y facilidades de la formulación de interrogantes complejos. Los enfoques basados en conocimientos fueron introducidos con el objeto de permitir la representación y manipulación explícita de los diversos conocimientos disponibles, del contenido de la escena y de las propiedades de los operadores a aplicar o de las estrategias de exploración e interpretación a utilizar.

A continuación se hace una lista de los SBC que han sido implementados para apoyar las tareas de procesamiento y descripción de imágenes:

1. Sistemas Basados en Análisis Detallado de Algoritmos de Procesamiento de Imágenes
    a. Sistemas de Segmentación de Imágenes Basados en Reglas de Producción
    b. Sistemas de Segmentación de Imágenes Guiados por Objetivos
2. Sistemas de Supervisión de Programas
    a. Sistemas de Consulta
    b. Sistemas de Creación de Programas a Partir de Librerías
    c. Especificación Interactiva
    d. Especificación Mediante Comandos Abstractos
    e. Especificación Basada en Ejemplos
3. Sistemas de Integración Semántica de Programas
4. Sistemas de Comprensión de Imágenes Basados en Conocimientos
5. Sistemas Separados de Comprensión de Imágenes
6. Sistemas Híbridos de Comprensión de Imágenes

4. ARQUITECTURAS DE SISTEMAS DE VISIÓN DE PROPÓSITO GENERAL (SVPG)

Un Sistema de Visión de Propósito General (SVPG) es un sistema de dominio independiente y abierto que está disponible para construir, mantener y usar una representación interna de el mundo externo de información suministrada por medio de sensores físicos tales como cámaras de vídeo. Estas representaciones son utilizadas en otros sistemas que actúan en el mismo mundo externo pero careciendo de capacidades de percepción. Tales sistemas pueden interactuar con sistemas de visión a través de intercambio de información o de objetivos que los sistemas de visión deben satisfacer. Su actividad interna es realizada por el encadenamiento de procesos específicos en el contenido o en colaboración con otros sistemas con quienes él interactúa.

Se describe a continuación un SVPG basado en una aproximación de la Inteligencia Artificial Distribuida (IAD) [6], el cual utiliza agentes cooperantes (módulos inteligentes con capacidad de comunicación) conocido como Visión Como Proceso (VCP) [1].

Los sistemas de Visión Como Proceso están enfocados hacia las técnicas de control de los procesos de percepción en un sistema de visión integrado, figura 2. El mayor esfuerzo de estos sistemas radica en la fase de integración de los módulos o subsistemas. La característica principal de estos sistemas es la de activar subsistemas como los de operación de cámaras móviles en un entorno dinámico bajo las restricciones de tiempo real, teniendo un mayor efecto sobre el control del sistema. El sistema se compone de los siguientes agentes: Unidad de control de la cámara, módulo de descripción de imagen operando en múltiples resoluciones, módulo para extractar descripción en 3D de escenas de una secuencia de imágenes, procesos para mantener dinámicamente una descripción simbólica de una escena utilizando información de otros procesos y de un conocimiento a priori acerca de la escena.

- **Divisiones Horizontales:** Los sistemas VCP utilizan niveles explícitos de representación. Los cuatro niveles son: Imagen, Descripción 2D de la imagen, Descripción 3D de la imagen e Interpretación simbólica de la escena.
- **Divisiones Verticales**: Un agente focalizador está definido como la unión de unidades de procesamiento que transforman una sucesiva representación de una Región De Interés (RDI) dada.
- **Agentes Básicos:** En cada momento del procesamiento un agente básico puede ser definido dinámicamente. En este nivel el agente está afectado por la RDI del correspondiente nivel de representación. De esta manera se obtiene el mismo nivel para una definición dinámica de varios agentes básicos homogéneos correspondiente a las varias RDI que se puedan definir.
- **Comportamiento:** El sistema tiene un comportamiento definido: su actividad está dirigida hacia la satisfacción de objetivos dados por el supervisor.

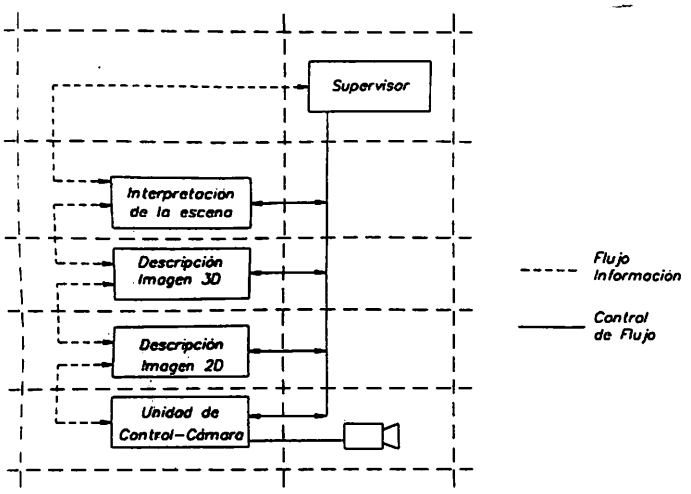


Figura 2. Arquitectura del Sistema VCP (Visión Como Proceso).

- **Estructura Interna:** El funcionamiento del VCP está basado en el ciclo Emparejar- Actualizar - Predecir, figura 3, que se adiciona a un ciclo de predicción y verificación. El ciclo incorpora una predicción temporal que dispone el sistema para integrar sus resultados en el tiempo, en el nivel de entrada el agente tiene una función de transformación que hace posible ir de un nivel de representación a un nivel superior. En el nivel de salida la fase de verificación es acompañada de una función de proyección que hace posible ir de una representación a otra de bajo nivel. Como consecuencia cada agente está en disposición de transformar la representación de un nivel a otro.

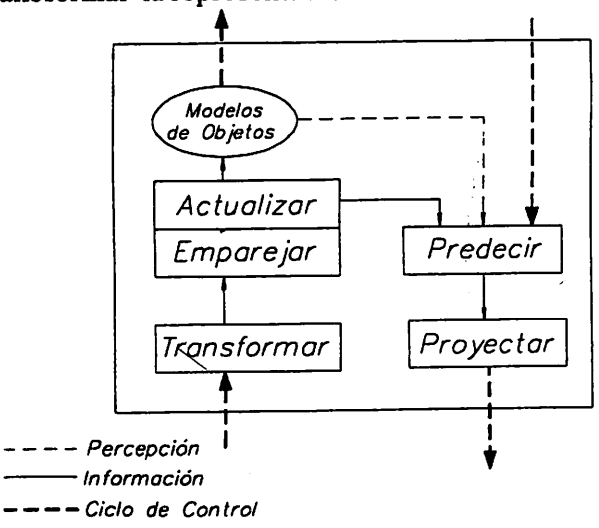


Figura 3. Modelo de agente del sistema VCP (Visión Como Proceso).

5. ESTRUCTURA DE UN SISTEMA BASADO EN CONOCIMIENTOS PARA EL RECONOCIMIENTO DE ELEMENTOS MECÁNICOS - SIBACORE

Para el diseño del Sistema Basado en Conocimientos para el Reconocimiento de Elementos Mecánicos se ha tenido en cuenta:

- Simplicidad en el sistema
- Utilización de software existente
- Modularización que permite la definición de subsistemas
- Manejo de representaciones en 2D
- Reducción de elementos mecánicos a: arandelas, tornillos y engranajes. Los cuales son representativos de los diversos niveles de complejidad en el reconocimiento.
- Conocimientos integrados a cada modulo de reconocimiento.

En la figura 4 se da la estructura del modelo SIBACORE:

- **Módulo de Reconocimiento:** Está compuesto de cuatro submódulos de reconocimiento de elementos mecánicos, a saber: submódulo de reconocimiento de tornillos, submódulo de reconocimiento de arandelas, submódulo de reconocimiento de engranajes y submódulo de reconocimiento total. Estos submódulos son explicados en detalle a continuación:

  - **Submódulo de Reconocimiento de Tornillos:** Los tornillos a reconocer son los normalizados. En la figura 5, se representa un tornillo de cabeza hexágona, en los cuales las dimensiones paramétricas se conocen a priori y la única dimensión que puede variar aleatoriamente es el largo del tornillo. Por lo tanto determinando la dimensión de la cabeza del tornillo o la dimensión del diámetro del tornillo este queda completamente determinado.

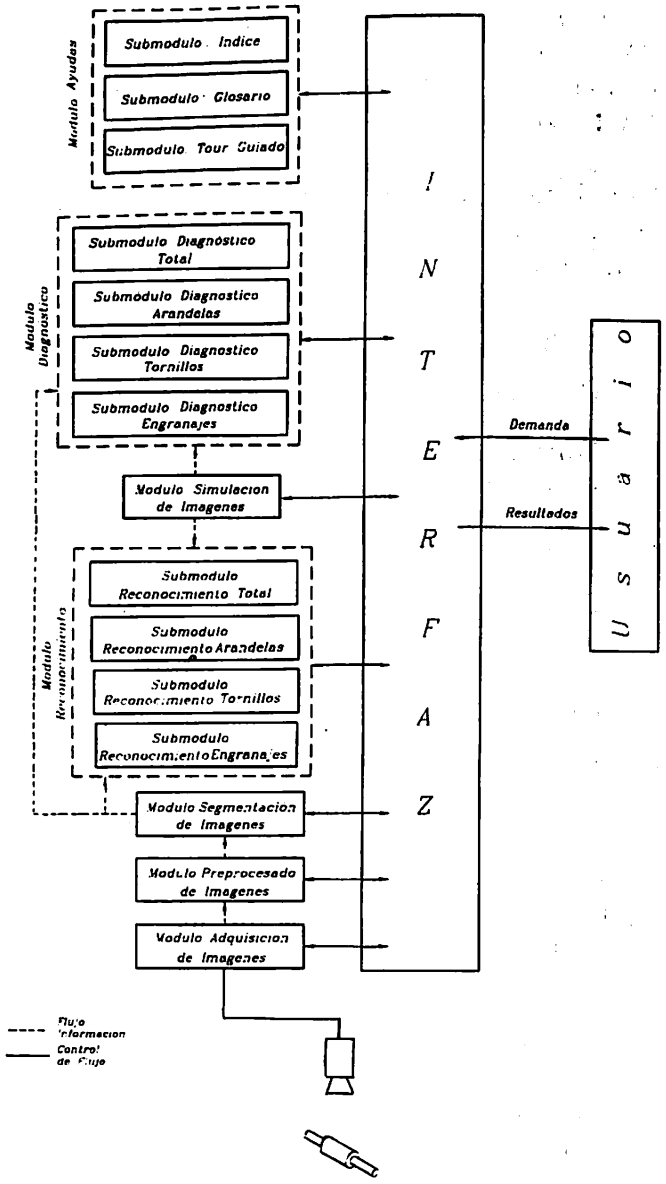


Figura 4 Estructura para un Sistema Basado en Conocimientos Para el Reconocimiento de Elementos Mecánicos (SIBACORE).

- **Submódulo de Reconocimiento de Arandelas:** Contiene los algoritmos para encontrar las arandelas sobre una imagen desplegada en la pantalla de computador.

- **Submódulo de Reconocimiento de Engranajes:** La parte exterior de un engranaje, conformado de los dientes, queda completamente determinada si se sabe el módulo, el número de dientes y el ángulo de presión. Por lo tanto el reconocimiento a partir de la imagen en la pantalla gráfica se hará: encontrando el centro del engranaje, determinando el tamaño del diente (mediante determinación del radio de adendo y dedendo) y calculando el número de dientes.

- **Submódulo de Reconocimiento Total:** Este sistema permite encontrar cada uno de los elementos mecánicos existentes en la imagen. Recorre la imagen, ejecutando automáticamente los tres submódulos descritos anteriormente. Por la eficiencia de los módulos se ejecuta primero el de reconocimiento de arandelas, luego el de tornillos y finalmente el de engranajes.

- **Módulo de Diagnóstico:** El Módulo de Diagnóstico debe reconocer las piezas defectuosas y especificar cuales son los defectos detectados y su posible arreglo. Se ha creado un módulo de diagnóstico independiente del reconocimiento Este módulo está compuesto por los tres subsistemas siguientes:

  - **Submódulo Diagnóstico de Arandelas:** El reconocimiento se basa en el principio de que una arandela en imagen binaria son dos círculos concéntricos. Pero ahora cualquiera de los dos círculos puede ser defectuoso, y el diagnóstico debe indicar cual es el círculo defectuoso y ubicar la localización del imperfecto para su arreglo. Se indicará si el defecto es por exceso o disminución del radio.

  - **Submódulo Diagnóstico de Tornillos:** De los posibles tornillos defectuosos a reconocer se han clasificado los siguientes: Tornillos con cabeza seccionada, el vástago del tornillo torcido y rosca del tornillo eliminada (rosca "robada"). Al igual que en el módulo anterior se diagnosticará el defecto y como arreglar su imperfección.

  - **Submódulo Diagnóstico de Engranajes:** El defecto más común en los engranajes es la perdida de uno o varios dientes. Éste módulo

de diagnóstico dirá cuántos dientes faltan y en que posición irían.

- **Submódulo Diagnóstico Total:** Este sistema permite encontrar cada uno de los elementos mecánicos defectuosos existentes en la imagen. Recorre la imagen, ejecutando automáticamente los tres submódulos descritos anteriormente. En orden de complejidad ascendente se ejecuta primero el diagnóstico de arandelas, luego el de tornillos y finalmente el de engranajes.

Módulo de Simulación de Imágenes: Este módulo puede reemplazar los tres primeros módulos: adquisición, preprocesado y segmentación de imágenes. Se encarga de entregar una imagen binaria con el contorno ideal o defectuoso de la pieza. Los contornos simulados corresponden a arandelas, a un conjunto de tornillos normalizados de cabeza hexagonal y a engranajes definidos básicamente por el módulo.

Módulo de Ayudas: Muy importante en cualquier software actualmente son las ayudas que facilitan la utilización y comprensión del mismo. Las ayudas implementadas en SIBACORE se compone de los siguientes submódulos:

- **Submódulo Indice:** Permite acceder a la definición de cada una de las ordenes del sistema, al igual que las definiciones de cada una de las partes constitutivas del sistema.

- **Submódulo Glosario:** Aquí se encuentran las definiciones mas frecuentemente usadas en el ambiente de la visión artificial. Su utilización se hará a través de la interfaz, al igual que los demás submódulos que componen el sistema.

- **Submódulo Tour Guiado:** Para facilitar el uso del paquete se crea este submódulo. De acuerdo a la arquitectura del sistema se tienen secciones de enseñanza, donde a través de ejemplos se ilustra la manera de utilizar cada uno de los menús que componen el sistema.

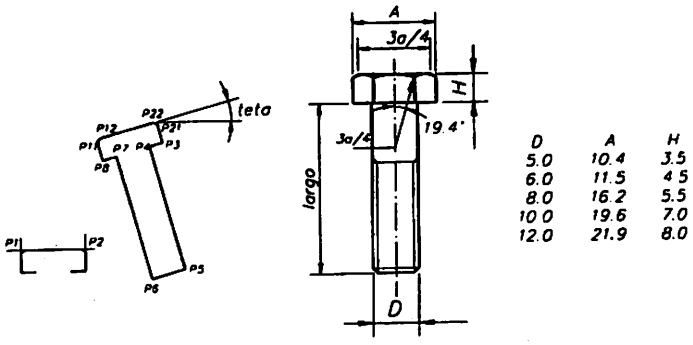
- **Interfaz Usuario / Sistema:** Permite al usuario interactuar con cada uno de los módulos de SIBACORE antes descritos. Lo anterior con el fin de ejercer un control sobre la ejecución de cada uno de los algoritmos que llevan a cabo las tareas que el proceso de reconocimiento requiere. Hay procesos que son secuenciales y por lo tanto se han de ejecutar en el orden previsto.

6. ALGORITMOS PARA EL RECONOCIMIENTO DE TORNILLOS

El reconocimiento de de tornillos esta basado en la posibilidad de detectar líneas rectas. Como se ve en la figura 5, en las imágenes binarias de los contornos de estos elementos mecánicos predominan los círculos y las líneas rectas.

Algoritmo de Bresenham Para Representar Líneas Rectas: La representación de líneas rectas en la pantalla gráfica, se hace frecuentemente utilizando el Algoritmo de Bresenham, ya que es un método eficiente para convertir por rastreo líneas rectas, pues solo utiliza adición, sustracción y multiplicación de enteros por 2. El método funciona de la siguiente manera [7]:

- Sea la línea de la figura 6, la que se desea convertir por rastreo. Luego la mejor aproximación a la recta verdadera esta descrita por los pixeles que queden a la menor distancia de la línea verdadera



D	A	H
5.0	10.4	3.5
6.0	11.5	4.5
8.0	16.2	5.5
10.0	19.6	7.0
12.0	21.9	8.0

Figura 5. Parámetros que determinan un tornillo de cabeza hexagonal y puntos claves en la determinación de su contorno respectivo.

Sea t_i la distancia de los pixeles que quedan inmediatamente por encima de la línea verdadera y s_i la distancia de los pixeles inmediatamente por debajo.

- La variable de decisión $d_i = s_i - t_i$, define cual es el pixel mas cercano a la línea verdadera. Si d_i es menor de cero, el pixel más cercano es el de abajo y si d_i es mayor o igual a cero, el pixel mas cercano será el de por encima.

- Los valores d_i se calculan así: $d_1 = 2dy-dx$ donde:

$$dy = y_2 - y_1$$
$$dx = x_2 - x_1$$

Si $d_i \geq 0$ Entonces : $x_i+1 = x_i + 1$, $y_i+1 = y_i + 1$, $d_i+1 = d_i + 2(dy-dx)$

Si $di < 0$ Entonces : $x_i+1 = x_i + 1$, $d_i+1 = d_i + 2dy$

- Esta técnica se utiliza para líneas rectas cuya pendiente sea menor de 45°, para pendientes mayores de 45° y menores de 90°, tan solo se ha de invertir x por y para efectuar los cálculos y luego volver a intercambiar para realizar el gráfico.

$$s_1 = (dy/dx)X_1 - Y_i$$
$$t_i = 1 - (dy/dx)X_i + Y_i$$
$$d_1 = s_i - t_i = 2(dy/dx)X_1 - 2y - 1$$

Redefiniendo d_1 para el computo:

$$\boxed{d_i = 2X_i\, dy - 2y_i\, dx - dx}$$

Para d_1 $\begin{cases} X_1 = 1 \\ Y_1 = 0 \end{cases}$

$$d_1 = 2dy - dx$$

Si $d < 0$ Solo se incrementa X

$$X_{i+1} = X_i + 1 \quad ; \quad Y_{i+1} = Y_1$$
$$d_{i+1} = 2(X_i + 1)dy - 2y_1\, dx - dx$$
$$\boxed{d_{i+1} = d_i + 2dy}$$

Si $d_i \geq 0$ Se incrementa X y Y

$$X_{i+1} = X_i + 1 \quad ; \quad Y_{i+1} = Y_i + 1$$
$$d_{i+1} = 2(X_i + 1)dy - 2(y_i + 1)dx - dx$$
$$\boxed{d_{i+1} = d_i + 2(dy - dx)}$$

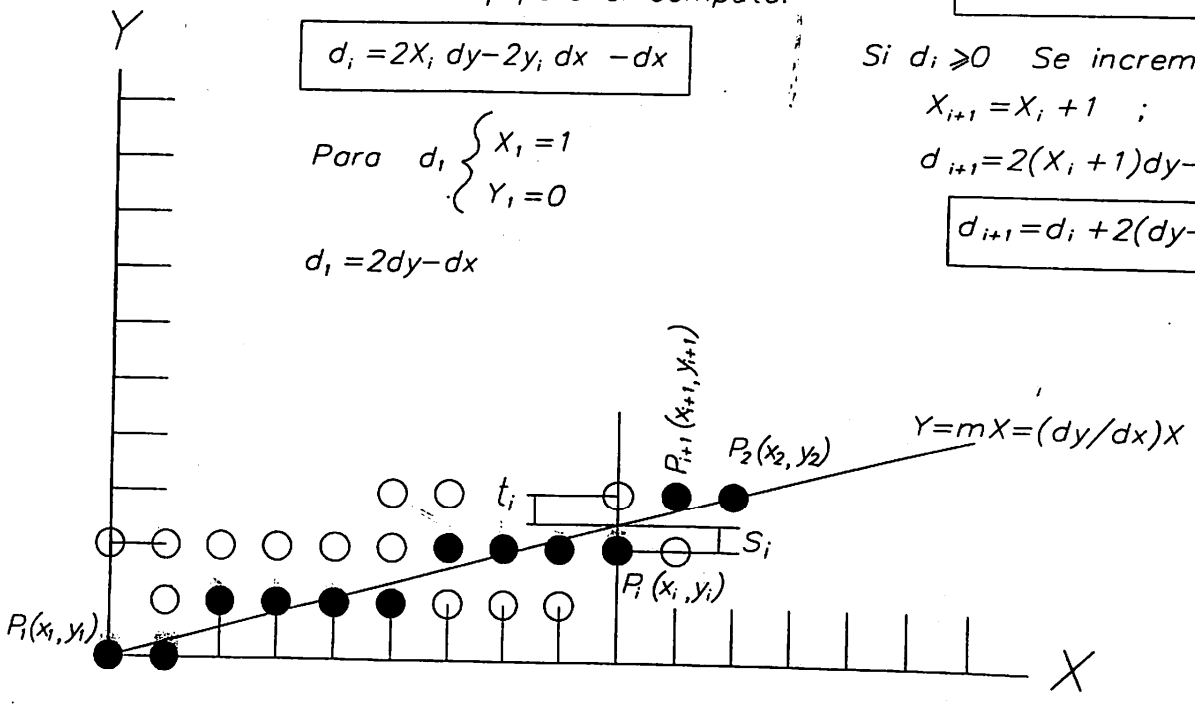


Figura 6. Representación de lineas rectas en la pantalla gráfica, utilizando el algoritmo de Bresenham. En la figura la recta tiene una pendiente entre 0° y 45°.

A continuación se presenta el algoritmo que permite representar una línea recta con pendiente m entre los valores; $0 \leq m \leq 1$, en la pantalla gráfica del computador:

Algoritmo Para Detectar Líneas Rectas: El Análisis de Nivel Intermedio, consiste en obtener información abstracta de las imágenes partiendo de las imágenes mismas. En este estado, ya no interesa convertir una imagen en otra, como sucedió en el Procesamiento de Imágenes. En este nivel intermedio se trata de investigar sobre las posiciones y orientaciones de imágenes elaboradas. Se trata de encontrar formas geométricas sencillas sin importar que pueda representar, por ejemplo si un círculo existe en una parte de la imagen sin interesar si este es una rueda y pueda tener defectos. **[3].**

Tabla 1. Algoritmo para representar líneas rectas con pendiente $0 \leq m \leq 1$

Adquirir x_1, y_1, x_2, y_2*; Punto inicial y final de la línea recta*

$dy = y_2 - y_1$
$dx = x_2 - x_1$
$d1 = 2dy - dx$
$i = 1$
$x = x_1$
$y = y_1$
Dibuje pixel en x, y
repita
  Si $di \geq 0$
    $x = x + 1$
    $y = y + 1$
    $d_i+1 = d_i + 2(dy - dx)$
  De lo contrario
    $x = x + 1$
    $d_i+1 = d_i + 2dy$
  Dibuje pixel en x, y
hasta que $x > x_2$

Si, partimos del hecho de tener una imagen binaria que representa los bordes del objeto, fondo blanco y contornos en negro, y si empezamos a realizar un rastreo vertical a lo largo de la pantalla, se encontrará con las posibilidades de existir líneas rectas como se muestra en la figura 7. El efecto de segmentado en las líneas se debe a que la pantalla gráfica del computador esta compuesta de pixeles, y esta es la manera en que se pueden representar líneas rectas en la pantalla gráfica.

Desde los años 60, se ha utilizado la Transformada de Hough como el método mas importante para detectar Líneas Rectas. En este artículo se ilustrará el método implementado en SIBACORE para detectar Líneas Rectas basado en la definición de las líneas rectas en la pantalla gráfica del computador y de la utilización de filtros de seguimiento en la pantalla gráfica.

En la figura 7, se muestran los varios tipos de línea recta que se encuentran en la pantalla gráfica, cuando se realiza un rastreo vertical de izquierda a derecha. Los varios tipos de línea recta se han clasificado de acuerdo a la pendiente de estas. Los filtros adecuados para verificar la existencia de una línea recta se muestran en la figura 8.

Con solo explicar el funcionamiento de uno de los cuatro filtros se entenderá el funcionamiento de los otros tres. El filtro uno, esta basado en que el pixel

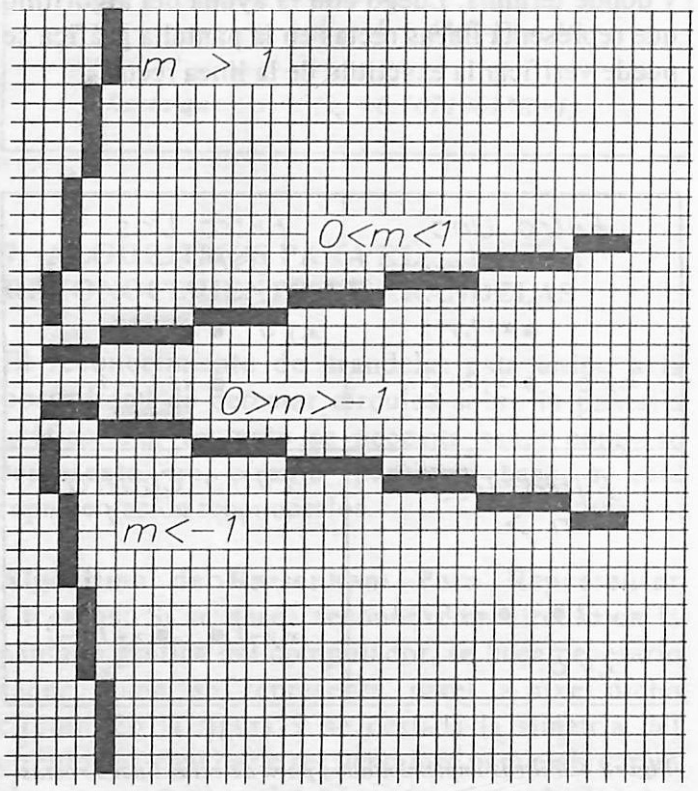


Figura 7. Líneas rectas posibles en un rastreo vertical y de izquierda a derecha en la pantalla gráfica. m es la pendiente de la recta.

en *x, y* puede ser parte de una línea recta, cuya pendiente esta entre 0 <= m <= 1, por lo tanto deberá existir un pixel a su derecha: *x+1, y* o a la derecha pero en la parte inmediatamente superior: *x+1, y+1*. Si el pixel existe en cualquiera de las dos posiciones, se repetirá el proceso hasta no encontrar pixeles, a partir del último encontrado.

El filtro nos dará el punto inicial y final de una posible línea recta, ahora tan solo se ha de verificar pixel a pixel su identidad con el algoritmo que define una línea recta, y así por comparación se podrá definir la exactitud de la línea recta detectada.

Las líneas detectadas como líneas rectas, se les puede ir cambiando de color para que el algoritmo sea mas eficiente. Si se desea eliminar ruido, líneas rectas inferiores a una longitud *n*, se grafican como líneas rectas con el color del fondo de la imagen.

El algoritmo que se da a continuación (ver tabla 2)permite saber donde empieza la posible línea recta y donde termina. Luego con la ayuda del algoritmo que representa líneas rectas en la pantalla gráfica se puede verificar la exactitud de la línea recta.

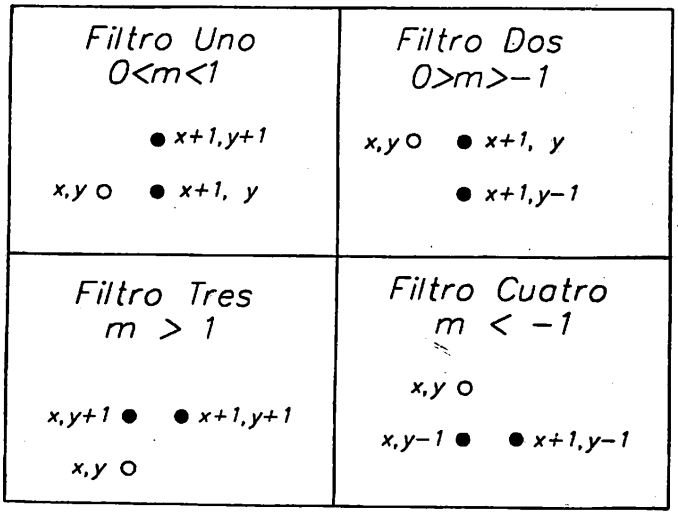


Figura 8. Filtros implementados para detectar líneas rectas según su pendiente

Tabla 2. Algoritmo para el filtro Uno, el cual reconoce los puntos inicial y final de una línea recta con pendiente $0 \leq m \leq 1$

```
x = 0 ;   y = 0
repita
  Si punto en x, y; Puede ser inicio de una línea
recta
        x1 = x
      y1 = y
      repita
            Si punto en (x+1, y)
          x = x+1
            De lo contrario
                  Si punto en (x+1, y+1)
                        x = x+1
                        y = y+1
            De lo contrario    ; Se llego al punto
final de la línea
                        x2 = x
                        y2 = y

 hasta que no haya punto en (x+1, y) ni en (x+1,
y+1)

    De lo contrario
    y  = y + 1
            Si y > 479
            y  = 0
                    x = x + 1
                  Dibuje pixel en x, y
    hasta que   x > 639
```

Con las herramientas definidas podemos pasar a describir el algoritmo que permite detectar tornillos en la pantalla gráfica:

Algoritmo Para Reconocer Tornillos: A continuación el algoritmo que reconoce tornillos sobre la pantalla gráfica:

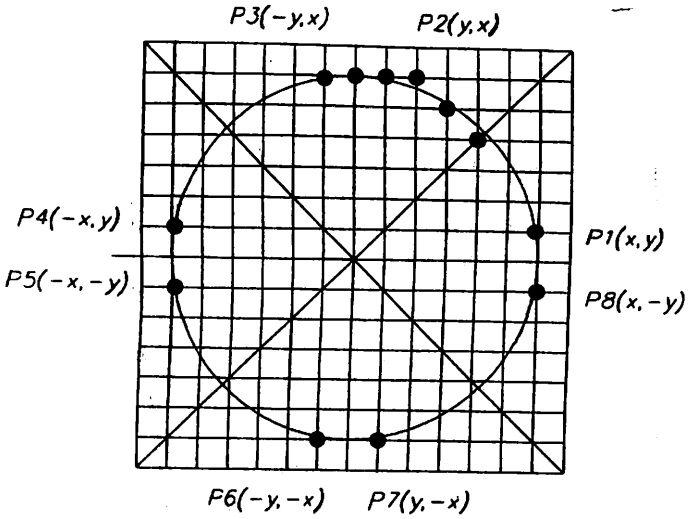


Figura 9. Círculo convertido por rastreo con el algoritmo de Bresenham, también se resalta la simetría del círculo en ocho direcciones.

Tabla 3. Algoritmo para detectar tornillos en una pantalla gráfica.

```
x = 0 ;  y = 0
Repita
    Si pixel blanco en x, y  Entonces ;
Puede ser inicio de una línea recta
    Aplicar filtros para determinar ptos. inicial y
final de la recta
    Determinar teta como ángulo de rotación
    Hacer A = 4* longitud de la línea/3
    Leer parámetros D y H correspondientes a A
            Calcular los puntos claves
            P11,P12,P21,P22 y P1 a P8
  Si existe pixel blanco en cada uno de los puntos
anteriores y hay líneas entre ellos
    Entonces
            Existe un tornillo
con las características A, D , H y largo P7 a P8
  Sino
    Siga rastreo de la pantalla en busca de
pixeles blancos
  FS
  Sino
            y = y + 1
                Si y > 479 Entonces
                        y = 0
                        x = x + 1
            FS
Hasta que  x > 639
```

Tabla 4. Algoritmo para representar círculos en la pantalla gráfica.

```
Adquirir xc, yc  ; Centro del círculo
Adquirir R       ; Radio del círculo
    x = xc  ;        y = yc -R
    di= 3-2*R
    repita
            E = x - xc  ;  F = yc-y
            Dibuje pixel en (xc+E, yc-F)  ;
Dibuje pixel en (xc+F, yc-E)

            Dibuje pixel en (xc+F, yc-E)  ;
Dibuje pixel en (xc+E, yc+F)

            Dibuje pixel en (xc-E, yc+F)
            Dibuje pixel en (xc-F, yc+E)
            Dibuje pixel en (xc-F, yc-E)
            Dibuje pixel en (xc-E, yc-F)
            Si  Di < 0
                    di = di + 4*E +6
                    x++;
            De lo contrario
                    di = di + 4*(E-F) + 10
                    x++; y++
    hasta que  x <  (xc +0.7071067*R+1)
```

7. ALGORITMOS PARA EL RECONOCIMIENTO DE ARANDELAS

El reconocimiento de arandelas esta unido a la posibilidad de detectar círculos sobre la pantalla gráfica. Por lo tanto se necesita saber como se representa un círculo para en base a esta representación reconocerlos.

Algoritmo de Bresenham Para Representar Círculos: Si se desea reconocer un círculo en la pantalla gráfica del computador, se hace necesario saber como se representa pixel a pixel dicho círculo. En la figura 9 se destaca la simetría del círculo, lo cual ha de ser ventajoso para elaborar un algoritmo, pues solo es necesario calcular las coordenadas de un pixel y hallar los otras siete coordenadas por reflexión.

Algoritmo Para Reconocer Arandelas Basado en la Transformada de Hough: Se utilizará la transformada de Hough para encontrar el centro de las arandelas, la transformada de Hough se basa en transformar el arreglo de coordenadas f(x,y)= I (intensidad del pixel) en un arreglo g(x)= f1 y otro h(y)= f2 (f1 y f2 son las frecuencia en que x,y puede ser centro de simetría), a partir de este centro se determinan dos radios y por comparación de pixeles existentes se determina si existen los dos círculos concéntricos, que nos lleva a concluir que existe una arandela. A continuación en la tabla 5 se da el algoritmo que reconoce arandelas **[2]:**

Tabla 5. Algoritmo para reconocer arandelas basado en la transformada de Hough

Inicializar los arreglos votos_en_x[640], votos_en_y[376]; Ptos. medios entre dos puntos
Inicializar los arreglos centro_en_x[20], centro_en_y[20]; Coordenadas con mayor votación de puntos medios. Posibles centros
mientras *encuentre_pto_hor(x,y) ; Rastreo horizontal hasta encontrar un pixel blanco*
lon= longitud al proximo pixel blanco o longitud hasta donde hay pixeles blancos
x = x + lon/2
votos_en_x[x]++
; Hasta barrer toda la pantalla
mientras *encuentre_pto_ver(x,y) ; Rastreo vertical hasta encontrar un pixel blanco*
lon= longitud al proximo pixel blanco o longitud hasta donde hay pixeles blancos
y = y + lon/2
votos_en_y[y]++
; Hasta barrer toda la pantalla
Seleccionar del arreglo votos_en_x[x], los que han obtenido mayoria de votos, mas de N, y guardar los valores x en el arreglo centros_en_x[]

Seleccionar del arreglo votos_en_y[y], los que han obtenido mayoria de votos, mas de N, y guardar los valores yx en el arreglo centros_en_x[]

repita
Combinar los valores de los arreglos centros_en_x[i], centros_en_y[j]

Tabla 5. Algoritmo para reconocer arandelas basado en la transformada de Hough (continuación)

R1 = longitud al proximo pixel blanco a partir de centros_en_x[i], centros_en_y[j] R2 = R1 + longitud al proximo pixel blanco a partir de centros_en_x[i]+R1, centros_en_y[j]

Si *Hay circulos con centro en centros_en_x[i], centros_en_y[j] y radios R1 y R2*
Se encontró una arandela
De lo contrario
Confinuar
hasta que Se agoten los arreglos *centros_en_x[i], centros_en_y[j]*

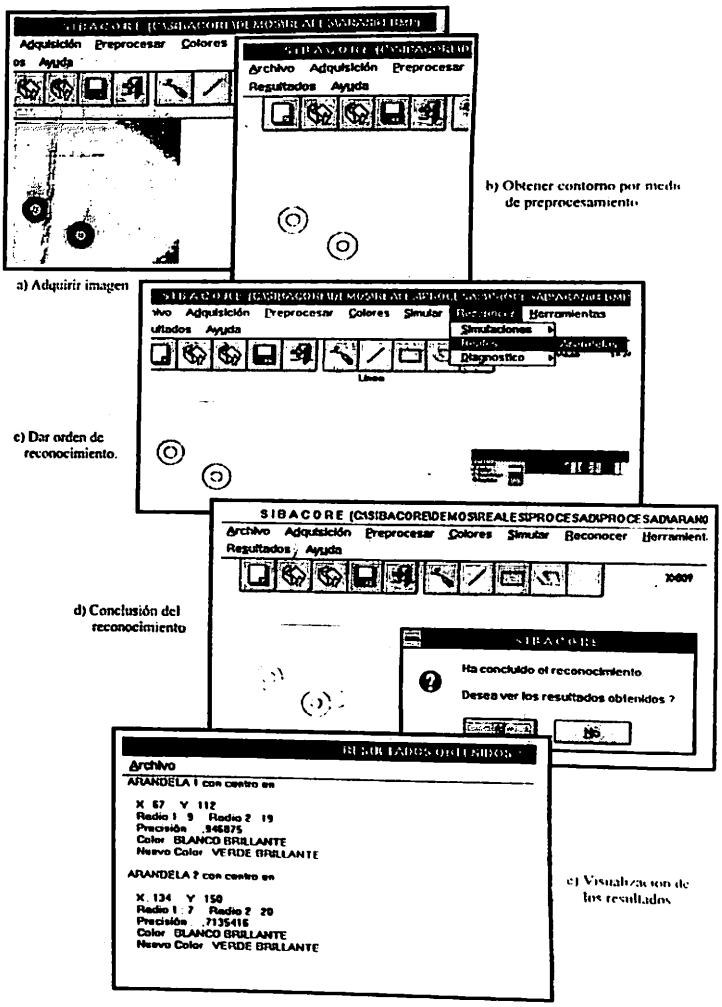


Figura 10. Reconocimiento de elementos mecánicos sobre contornos de piezas obtenidos por preprocesamiento

8. IMPLEMENTACIÓN DEL PROTOTIPO

En la figura 10 se muestra el prototipo implementado en funcionamiento. Se destaca las etapas de obtención de imágenes, el preprocesamiento, el reconocimiento y la emisión de los resultados. Los resultados se dan en forma gráfica y en forma de texto.

9. CONCLUSIONES

El artículo presentado tiene como objetivo mostrar la viabilidad de tener un Sistema de Visión Artificial que permita la identificación de Elementos Mecánicos, mediante la utilización de aplicaciones existentes para la adquisición y preprocesado de imágenes y complementando mediante desarrollos de programas para el reconocimiento y diagnóstico de imágenes. Resultó de gran interés, tener un módulo que permitió reproducir contornos de piezas defectuosas y no defectuosas, con lo cual se pudo evaluar los programas de reconocimiento y diagnóstico sin necesidad de tener que perfeccionar técnicas de preprocesado para obtener los contornos de las piezas.

No es fácil encontrar características comunes entre arandelas, tornillos y engranajes. En cada caso, los parámetros que definen la pieza son muy diferentes y en últimas se tiene un vector característico para cada individuo de la clase. Se requería de un sistema que permitiera la definición de objetos mediante abstracciones y que pudiera procesar las imágenes de la pantalla previamente adquiridas o simuladas.

Se debe profundizar en los siguientes temas para trabajos futuros, con el fin de mejorar el modelo:

1. Sistemas Basados en Conocimientos, que permitan supervisar y conducir los procesos de preprocesamiento, de acuerdo a los resultados de reconocimiento y diagnóstico. Se trata de obtener un proceso automático que permita mejorar la adquisición de imágenes mediante una mejor iluminación, relocalización de la cámara de video, calibración de la cámara, ajuste de parámetros para el preprocesamiento, etc., si el proceso de reconocimiento o diagnóstico no se puede llevar a cabo.
2. Procesos de Obtención de Contornos, que permitan obtener perfiles de las piezas con una resolución muy exacta. Las dificultades más comunes son:
   - Mejorar la definición o espesores de las líneas de contornos obtenidos en el preprocesamiento.
   - Aumentar la precisión en la definición de umbrales y fronteras de las piezas.
   - Definir técnicas de iluminación para mejorar los contornos obtenidos.

Proponer técnicas para obtener proyecciones ortogonales para piezas gruesas. Las imágenes obtenidas sobre estas piezas normalmente son perspectivas que dificultan el manejo de los objetos.

10. REFERENCIAS BIBLIOGRÁFICAS

[1] **BOISSIER Olivier y DEMAZEAU Yves.** A Distribuited Artificial Intelligence View on General Purpose Vision Systems.en MAAMAW 1991, pp 1-19, Laboratoire LIFIA/IMAG, Grenoble France.

[2] **CAÑÓN Jairo.** Modelo y Metodología de Vision Artificial para la Identificación y Diagnóstico de Elementos de Máquinas Mecánicas. Tesis de Maestría en Ingeniería de Sistemas, Universidad Nacional de Colombia - Sede Medellín, 1997.

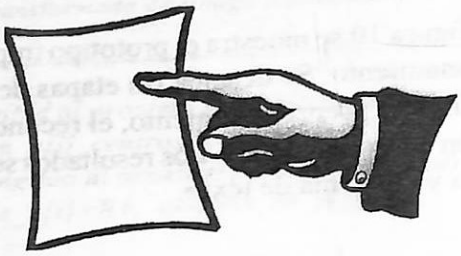
[3] **DAVIES E. R.** Machine Vision: Theory, Algorithms, Practicalities. Academic Press, pp 546., London 1990

[4] **FU K. S., GONZÁLEZ R. C. Y LEE C. S. G.** Robótica: Control, detección, visión e inteligencia McGraw-Hill, pp 599., 1990

[5] **MARAVALL GÓMEZ-ALLENDE** Dario Reconocimiento de Formas y Visión Artificial. Addison -Wesley Iberoamericana, Serie Paradigma, pp 433., 1994

[6] **OVALLE CARRANZA** Demetrio Arturo, **CAÑÓN RODRÍGUEZ** Jairo, Sistemas Inteligentes Artificiales y su Aplicación en Ingeniería, Editorial Universidad Nacional de Colombia, Santafé de Bogotá, pp. 367, 1997.

[7] **PLASTOCK** Roy A. y **KALLEY** Gordon Gráficas Por Computador McGraw-Hill, Serie de Compendios Schaum, pp 348., 1987



Si desea solicitar información Adicional sobre los artículos de esta revista, envíenos por correo o fax el desprendible que encontrará al final.