Detección en tiempo real de fallos en piezas industriales con simetría de revolución

Gustavo Sutter
INCA Investigación en Computación Aplicada
Facultad de Ciencias Exactas
Universidad Nacional del Centro de la Prov. de Bs As
Tandil – Argentina
gsutter@ exa.unicen.edu.ar

Joan Aranda
Dep. d'Enginyeria de Sistemes Automàtica i Informàtica Industrial (ESAII)
Universitat Politècnica de Catalunya
Barcelona – Espanya

Resumen

Se describe una aplicación para la detección de fallos en el ensamblado de muelles (resortes), utilizando la transformada polar discreta. La aplicación utiliza algoritmos de visión por ordenador. La velocidad de operación es crítica, dado que la respuesta debe ser dada en tiempo real. Los problemas a detectar son los defectos en uniones, la excentricidad de la pieza y estiramientos en el muelle. Se propone un algoritmo sencillo y eficiente, más la generalización de esta técnica para otros problemas similares.

Palabras Claves: Visión por ordenador, transformada polar discreta, imágenes.

Definición del problema:

Se pretende determinar las piezas con fallos en una cadena de producción a fin de evitar el ensamblado de estas con otras piezas de mayor complejidad. El elemento a ser examinado es un muelle de acero de unos 1 mm de diámetro, el que se dobla para formar una circunferencia (figura 1). Se cuenta con dos medidas de muelles para las pruebas (20 y 35 mm de diámetro), aunque se pretende dar una solución de forma genérica a piezas de otros diámetros . Los extremos del muelle se unen para formar la circunferencia y es en esta unión donde surgen la mayor cantidad de inconvenientes, ya que por problemas en el proceso de fabricación esta unión puede ser defectuosa. Además es deseable que la pieza no exceda de cierta excentricidad. En resumen los problemas a detectar son:

1. Detección de estiramientos en el muelle (zonas de menor densidad de material)
2. Detección de fallos en la unión
3. Detección de excentricidad en la pieza terminada

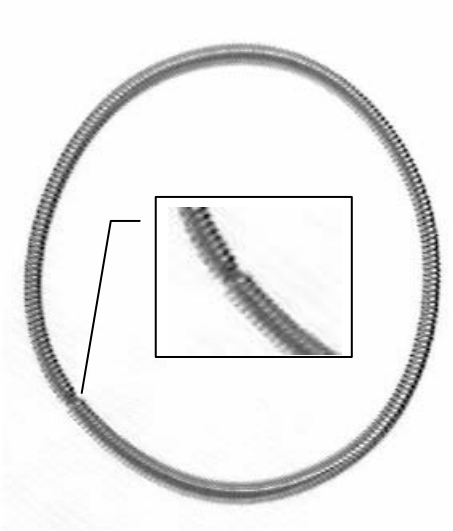


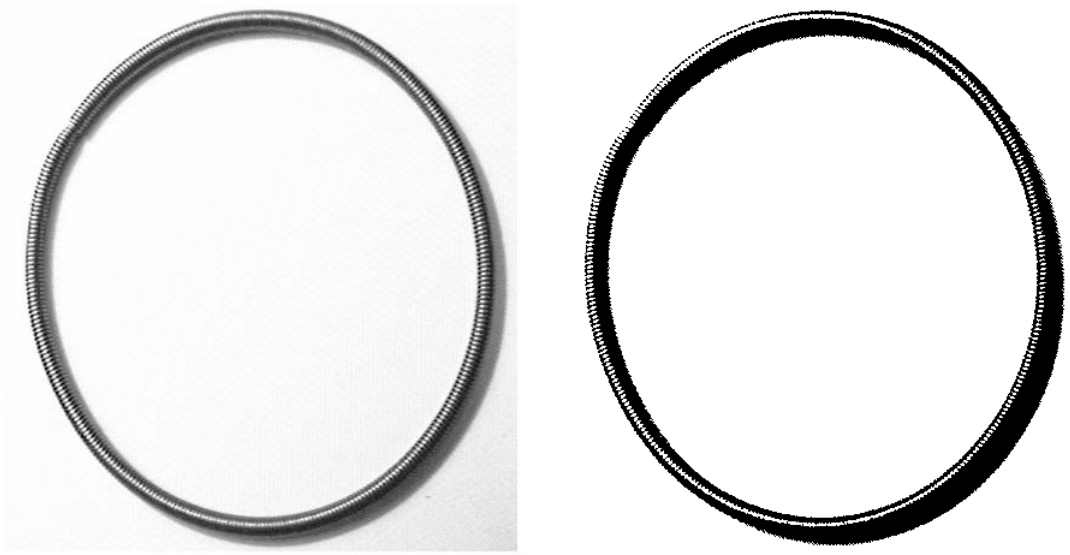
Figura 1. Pieza a ser examinada y detalle de la zona de unión

Adquisición, digitalización y binarización de la imagen.

La adquisición en tiempo real se podrá realizar con una cámara en tonos de grises (8 bits – 256 tonos) y una definición de 640x480 las que para esta aplicación serán suficientes. El aumento de definición aumentará el volumen de datos a procesar y consecuentemente el tiempo de proceso, lo que no se justifica respecto del incremento en la precisión del algoritmo.

Existen problemas con la iluminación, dada la superficie brillante de la pieza es fundamental trabajar sobre la iluminación de la pieza, eliminado los reflejos y evitando las sombras que distorsionarían por completo la imagen a analizar.

Tras las pruebas realizadas se ha visto la imperiosa necesidad de trabajar con luz difuminada a fin de evitar sombras como se explica en [3][4]. Se propone además, siempre que el proceso de producción lo permita, trabajar a contraluz. Con esta técnica de iluminación se eliminan los brillos y el objeto queda perfectamente delimitado y separado del fondo, facilitando por tanto el posterior proceso de binarización.



Figrura2. Imagen con deficiencias en la iluminación que provocan reflejos y sombras y la correspondiente imagen defectuosamente binarizada

En la figura 2 se puede observar una pieza con iluminación deficiente la que produce reflejos y sombras indeseables. En el proceso de binarización los reflejos son interpretados como puntos blancos (inexistencia de material) y las sombras como negros (material) deformando por completo la pieza a analizar.

Para las pruebas del algoritmo se utilizan imagines ya digitalizadas en ficheros en formato mapa de bits (Bit MaP - BMP) y tonos de grises. Las imágenes a evaluar se encuentran en un formatos de 8 bits (256 tonos de grises) yendo del 0 (negro) al 255 (blanco).

Tratándose de una imagen donde se conoce a priori que solo aparecerá un objeto de color oscuro sobre fondo claro (o viceversa) se propone binarizar la imagen adquirida a fin de comprimir su tamaño. Un menor volumen de información requerirá menor tiempo de proceso.

Para la binarización se propone aplicar una versión optimizada del algoritmo de valor único de Otsu [7]. No se han utilizado algoritmos más sofisticados, como los de relajación [5][6] debido a su elevado coste computacional. Inicialmente se pretendía calcular el umbral para cada pieza individual, pero en vista de los resultados no ha sido necesario. El umbral se determina a partir de los resultados de este algoritmo para una serie de imágenes de prueba. Una vez calculado será un parámetro fijo del programa con el consiguiente ahorro de tiempo de computación por pieza.

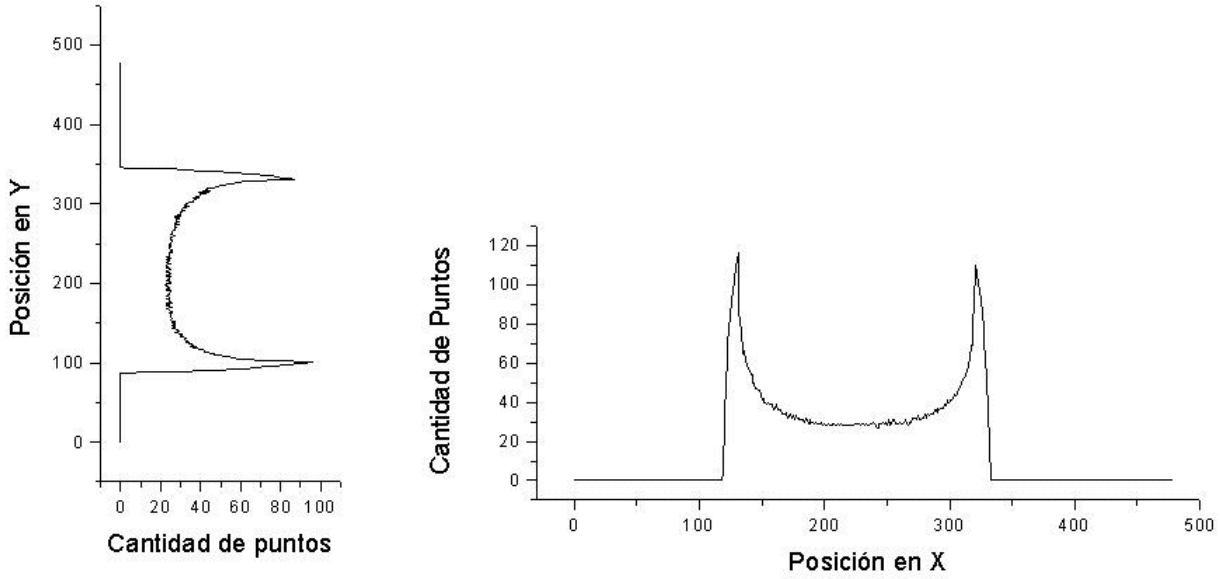


Figura 2. Proyección sobre los ejes Y y X respectivamente

Alternativas de implementación:

Dado los duros requerimientos temporales, el algoritmo desarrollado no puede ser demasiado complejo temporalmente. La primera aproximación es detectar sobre los histogramas de distribución de puntos sobre los ejes (figura 2) la posición de la unión, así como un eventual fallo en esta. La búsqueda de características atrayentes en este camino están fomentadas en el bajo costo de la generación de histogramas (una única pasada por cada píxel) Sobre los histogramas no se pudo obtener demasiada información útil. Dependiendo de la posición de la unión esta se vuelve indetectable. Además ciertas excentricidades pueden causar la perdida total de la información proveniente de la

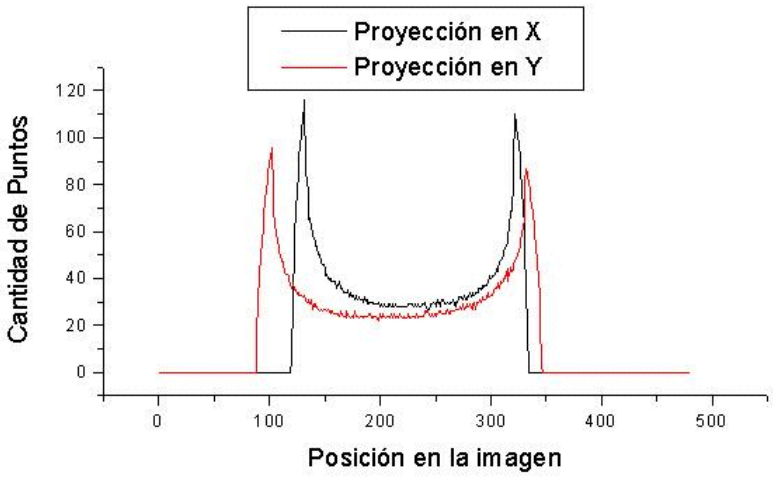


Figura 3. Comparación de las distribuciones

unión y disimular así defectos en ella. Una alternativa estudiada fue realizar además la proyección sobre otros ejes rotados respecto de los cartesianos X, Y (digamos X'e Y' rotados 45 grados) pero los resultados tampoco resultaron alentadores.

No obstante de la superposición de ambos histogramas se puede obtener la excentricidad [2] (uno de los parámetros para considerar defectuosa una pieza) pero muy difícilmente se pueda detectar una unión con problemas (figura 3).

Algoritmo Propuesto.

Este algoritmo esta basado en la transformada polar discreta de la imagen [1]. Este algoritmo posee una primera pasada para la detección del centro de la figura. A partir de ese punto se divide la circunferencia en N sectores circulares (figura 4) y se cuentan la cantidad de pixels en cada sector. La falta de puntos en un determinado sector o un descenso brusco en la cantidad de puntos indicará tanto la unión o bien un defecto. Existen 4 puntos a considerar:

a. La cantidad optima de sectores a utilizar
b. A partir de cuando un sector posee pocos puntos para considerarlo la unión
c. Cuando un sector indica un defecto.
d. La detección de la excentricidad

a. Cantidad Optima de Sectores

Se tomaron diferentes imágenes y se procedió al análisis de estas para diferente cantidad de sectores. En la figura 5 se puede observar para un muelle en particular el resultado de las diferentes divisiones. La utilización de potencia de dos surge útil a la hora de implementaciones en microcontroladores o sistemas dedicados, ya que la mayor complejidad del algoritmo esta en el calculo de la función arco tangente que en estos procesadores puede ser implementada como una tabla de lookup en RAM o ROM.

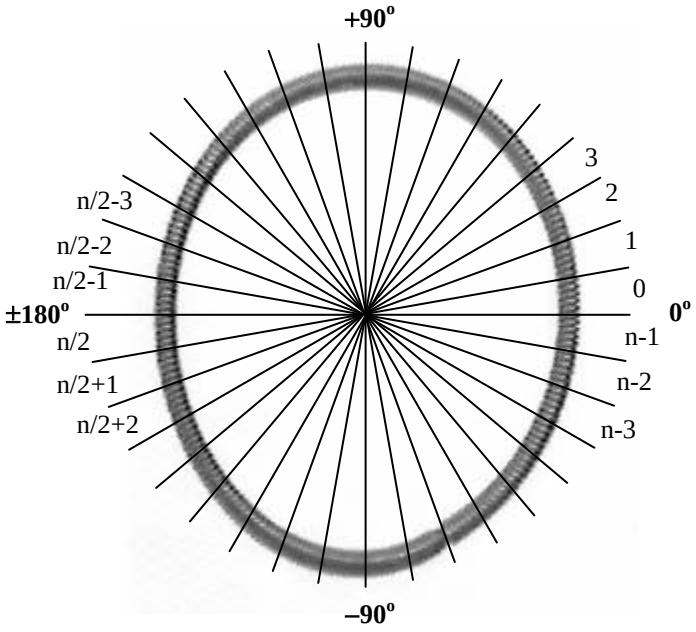


Figura 4. División de la circunferencia

La forma sinusoidal del histograma se debe a la excentricidad del resorte (un circulo daría una recta horizontal). Los picos indican la zona de mayor diámetro en tanto que los valles indican la zona de menor diámetro.
Como se puede ver en el gráfico cuando la cantidad de sectores es muy pequeña no se puede observar variación alguna y se corre el riesgo de disimular fallos (sobre todo si la división de zonas cae justo sobre la unión), en tanto si la cantidad es muy grande la variación de sector a sector es muy grande fundamentalmente debido a la finitud con

que se calcula la pertenencia de cada píxel a cada sector. Se estima optimo la utilización de 128 divisiones, en caso de grandes restricciones de espacio se podría trabajar con 64, pero sacrificando precisión y dando por erróneas piezas que quizás no lo sean.

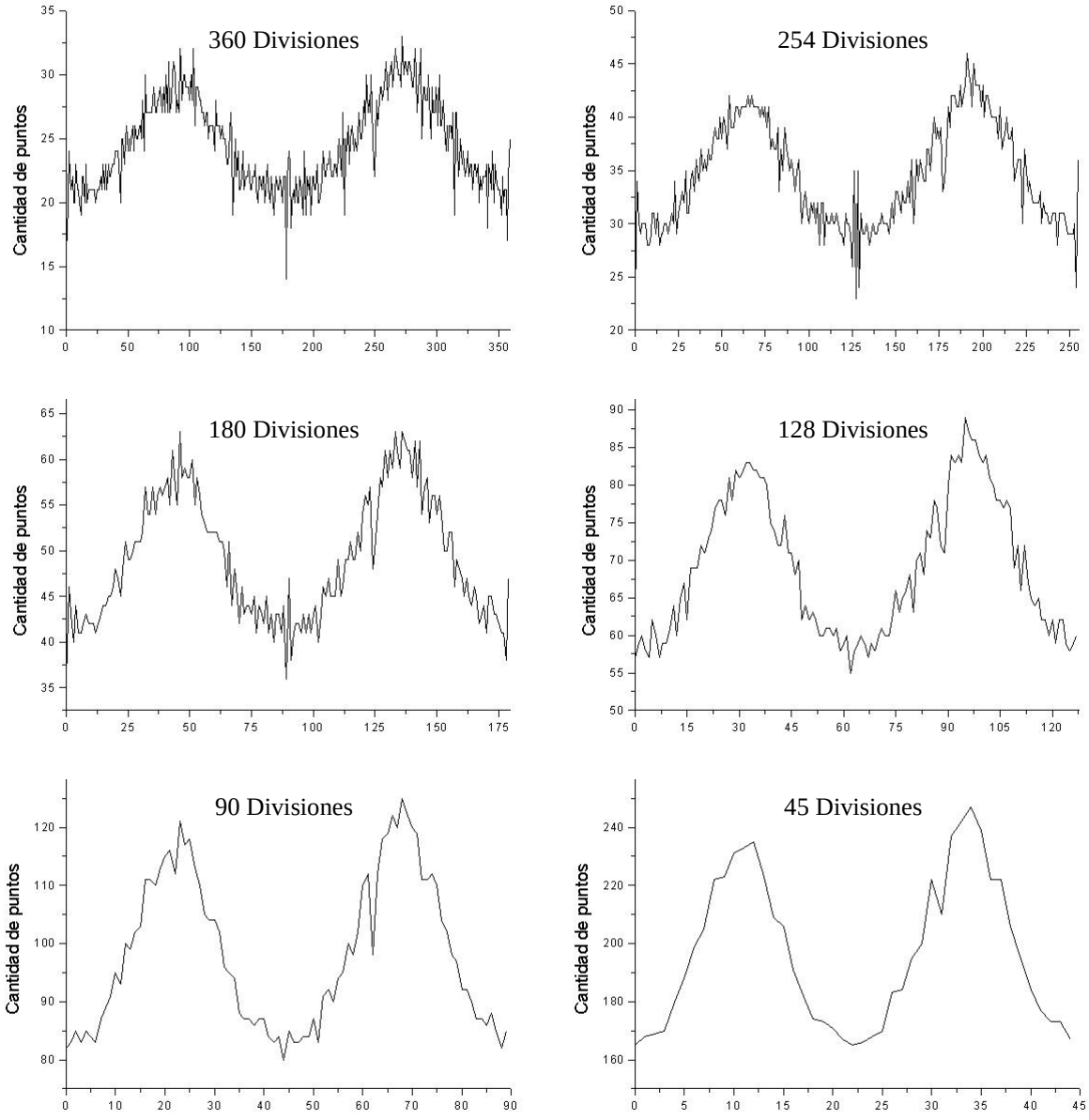


Figura 5. Diferente cantidad de sectores para la pieza de la figura 4.

b. Lugar de la unión del muelle.

Para un mejor entendimiento se procede a filtrar el histograma a fin de ver las variaciones en la cantidad de píxeles en cada sector, en la figura 6 se ve un ejemplo para el caso con 128 divisiones. Obsérvese que el mínimo valor corresponde a la posición donde se encuentra la unión.

La operación de filtrado se puede llevar a cabo mientras se construye el histograma. Para detectar esta máxima variación basta observar un punto a analizar y sus vecinos a izquierda y derecha. Pruebas empíricas muestran que utilizando una ventana de tamaño 5 (2 valores anteriores, los 2 posteriores y el propio) para 128 regiones logra muy buenos resultados. Utilizar mayor cantidad de regiones exige el aumento del tamaño de la ventana.

Cabe acotar que esta región de mínima cantidad de píxeles posee entre un 12% y 18% menos de píxeles respecto de la media de puntos de todas las regiones. Este es un punto que debe ser controlado al final de cada inspección para aceptar una pieza como buena.

c. Detección de fallas en la unión.

Tal lo expresado en el ultimo párrafo el sector mínimo se encuentra entre un 12% y 18% por debajo de la media, el hecho de detectar cualquier sector que se encuentre por debajo de este valor implica la calificación de defectuosa para la pieza.

En la figura 7 se puede observar una pieza con estas características y los histogramas correspondientes. Obsérvese la brusca disminución en la zona correspondiente al fallo. Existen no obstante fallos de menor tamaño pero también son detectados fácilmente.

d. Detección de excentricidad.

Si bien se pueden llevar a cabo mediciones más precisas, para esta aplicación basta con asegurarse que la distancia entre mínimos y máximos en los histogramas no excedan cierto valor. Es decir la con que la cantidad mínima no sea inferior a la cantidad máxima será suficiente para garantizar una excentricidad aceptable.

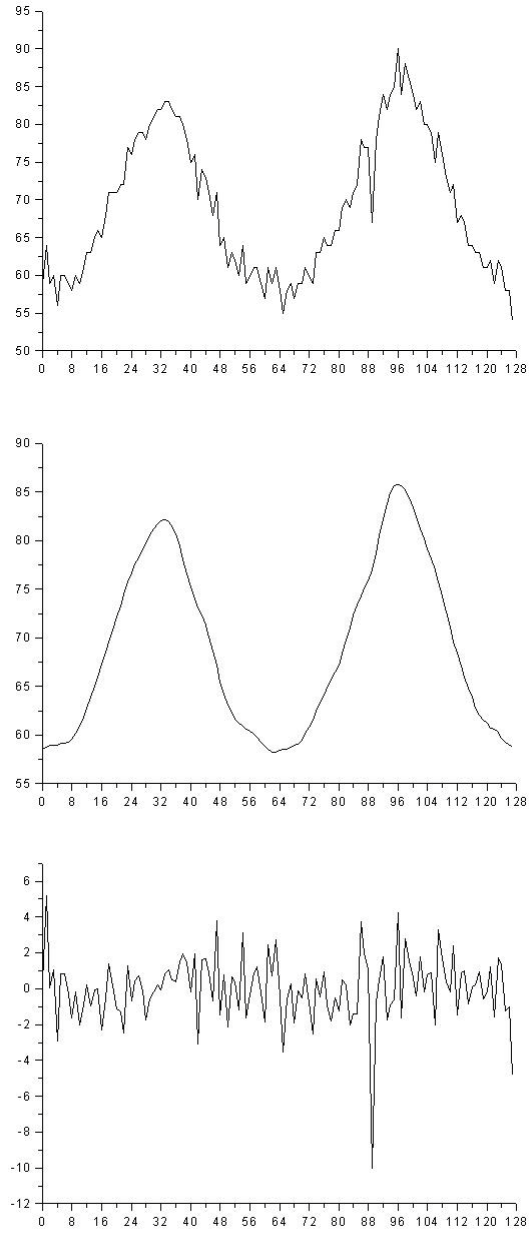


Figura 6. Histograma para 128 divisiones, promedio de 5 muestras usado como filtro e histograma filtrado.

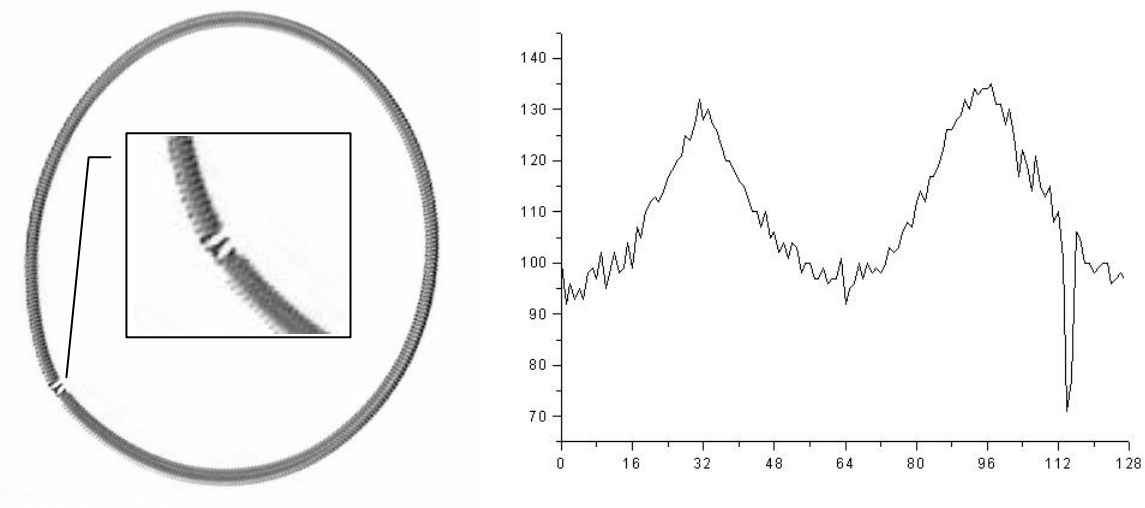


Figura 7. Pieza con problema de construcción, detalle de la zona con problemas e histograma.

Código de la aplicación

Se implemento un prototipo en el lenguaje de programación C compilando sobre MS Visual C++ y entorno Windows 9x/NT utilizando construcciones estándares ANSI de modo de permitir portabilidad a otros compiladores / plataformas.

Existen 2 ficheros con códigos: Archivo.c y Control.c, el primero implementa todas las funciones relacionadas con el tratamiento de ficheros, en tanto que el segundo implementa los algoritmos en si. Es de destacar que el primer fichero será reemplazado en el caso de trabajar con imágenes en tiempo real por otro que realice la adquisición de la imagen.

La salida del algoritmo en este prototipo son una serie de ficheros con información de la pieza analizada, en tanto en una aplicación real solo devolvería un valor entero indicando si es buena o defectuosa.

La forma de utilización desde la línea de comando es:

CONTROL nombreArchivo (sin extensión) [-cNRO] [-uNRO] [-a][-m][-b][-p][-y]

Donde:

-c	Cantidad Sectores a dividir la imagen (128 por defecto)
-u	Umbral del negro (220 por defecto)
-a	Imprimir el contenido de los secotres en archivo ´ NombreArchivo.XLS´
-m	Imprimir Métricas de la figura en Archivo ´NombreArchivo.dat´
-b	Generar imagen binarizada ´ NombreArchivo_b.bmp´
-p	Quita la información en pantalla
-y	Generar archivo con el contenido de las proyecciones ´ NombreArchivo_P.xls´

El pseudocodigo del algoritmo se puede resumir de la siguiente manera:

Binarización de la Imagen
Búsqueda del Centro de Gravedad
Cálculo de la distribución de píxeles por sectores.
Análisis de las variables para determinar la calidad de la pieza.

El primer paso de la binarización de la imagen puede ser realizado mientras se realiza el cálculo del centro de gravedad, dado que se reduce a comparar el valor de cada píxel (nivel de gris) con el umbral que determina que es blanco o negro. Aquí también se pueden calcular las proyecciones

```
//Busqueda del centro de gravedad, y proyecciones sobre los ejes X e Y

for (j=0; j < tamY; j++) {              // Recorro las filas
    for ( i=0; i < tamX; i++) {         // Recorro cada fila
        pos_x = j*tamX+i;               // Cálculo de la posición en la imagen
        if (imagen[pos_x] <= UMBRAL)    // Es un punto negro
              {totalPuntos++;
               ponderadoX += i;  ponderadoY += j;
              if (i > maxX) maxX = i; if (i < minX) minX = i;
              if (j > maxY) maxY = j; if (j < minY) minY = j;
 }  }         }
medX = (float)ponderadoX / totalPuntos; // punto medio en X
medY = (float)ponderadoY / totalPuntos; // punto medio en Y
```

Figura 8. Código para el cálculo del centro de la pieza, máximos y mínimos sobre los ejes

sobre los ejes y los valores máximos y mínimos en X e Y para el posterior análisis de excentricidad.

Para el cálculo de la distribución se utiliza la función arco tangente (atan), la que permite calcular el ángulo en que se ubica un punto respecto del centro de gravedad de la pieza (figura 9). En el código de la figura 10 se puede observar el cálculo de la cantidad de píxeles para cada sector, así como el cálculo para sectores desplazados medio sector. Esto último se llevo a cabo para evitar el caso límite ocasionado cuando una división de zona esta en medio de una falla disimulando este hecho. Las pruebas demostraron que utilizando la suficiente cantidad de divisiones (más de 64) esto deja de ser necesario.

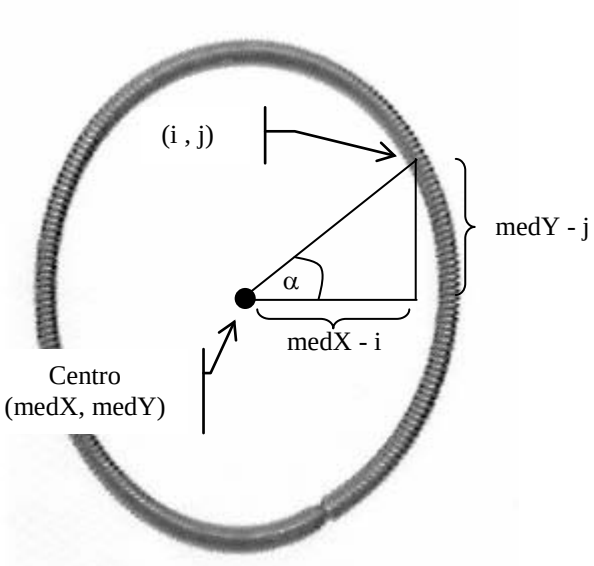


Figura 9. Calculo del ángulo respecto del centro

La función atan2(x,y) realiza la función arcotangente de los parámetros x, y, tiene la ventaja respecto de la función atan(x) que esta definida para cuando x e y son 0. Para una implementación en sistemas dedicados o para obtener más velocidad se podría pensar en una tabla (lookup table) que implemente dicha función. El tamaño de esta tabla tendría una tamaño de A^2 / 4 palabras de $\log_2 C$ bits cada una. Donde A es tamaño máximo en píxeles de la figura y C la cantidad de sectores. Para el caso de tener una figura de máximo 480 pixeles y 128 sectores se tendría una

memoria de 28800 palabras de 7 bits. Recordar que por propiedades trigonométricas basta con guardar los valores para un octavo de la circunferencia (45º o Π/4 radianes).

```
//determinación de la distribución en cada sector

for (j=0; j < tamY; j++) {              // Recorro las filas
    for ( i=0; i < tamX; i++) {         // Recorro cada fila
         pos_x = j*tamX+i;
         if (imagen[pos_x] <= UMBRAL)   //es un punto negro
             {float at = atan2((j-medY),(i-medX));
              float in = (at/PIx2)+0.5; //Normalización al intervalo [0,1]
              sectores[(int)(in*CANT_SECTORES)]++;
              sectores2[(int)(in*CANT_SECTORES+0.5)%CANT_SECTORES]++;
}     }      }
```

Figura 10. Código para la generación de la distribución

El análisis de las variables para determinar la calidad de la pieza se puede observar en la figura 11. En ella se destacan por un lado el error correspondiente a la falta de puntos en un sector y el calculo de excentricidad. Por otro lado debe ser también controlada la cantidad de puntos negros encontrados, con el objeto de detectar errores en el algoritmo de binarización. Valores fuera de los parámetros máximos y mínimos puede indicar una incorrecta elección del valor de umbral o una digitalización con demasiados ruidos que provocan que zonas obscuras sean interpretadas como parte de la pieza.

Los dos primeros pasos descriptos tiene un costo computacional O(n) con n cantidad de bits de la imagen, en tanto que el segundo es proporcional a la cantidad de sectores. Este entonces es un algoritmo de solo 2 pasadas sobre la imagen, lo que permitirá una operación a muy alta frecuencia.

```
// Análisis de las variables para determinar la calidad de la pieza.

int TAM_VENT_2 = TAM_VENT / 2;          //se supone siempre el tamaño impar
for (i=-TAM_VENT/2; i <= TAM_VENT/2; i++)
     media += sectores[(CANT_SECTORES + i)%CANT_SECTORES];
media /= TAM_VENT;
for (i=0; i < CANT_SECTORES; i++)
    {if ((sectores[i] - media) > TOLERANCIA_UNION) errorToleranciaUnion = 1;
     media += (sectores[(i+TAM_VENT_2)%CANT_SECTORES] -
               sectores[(i-TAM_VENT_2)%CANT_SECTORES]) / CANT_SECTORES;
    }
difX = maxX-minX; difY = maxY-minY;
long dif = (difX > difY) ?difX / difY : difY / difX;
if (dif > TOLERANCIA_EXEN) errorToleranciaExen = 1;
if ((totalPuntos > MAX_PIXEL)&&(totalPuntos < MIN_PIXEL)) errorPixeles = 1;
```

Figura 11. Código para la detección de fallas en las piezas

El tiempo de corrida es dependiente de la plataforma objetivo. Fuera de la etapa de adquisición de datos y escritura en ficheros el algoritmo tarda sobre un ordenador Pentium 200 Mhz, entorno Windows 95 en el orden de 0,11 segundos. Sobre un Pentium II Móvil a 366 Mhz y Windows 98 0,05 segundos y en tanto que en un Pentium III 450 Mhz y windows NT 4.0 unos 27 milisegundos. Existen no obstante varias optimizaciones dependientes e independientes de la arquitectura que se pueden llevar a cabo para mejorar estos tiempos.

Otras Aplicaciones

La utilización de la transformada polar discreta en la detección de fallos en piezas con simetría de revolución es muy atractiva dado su bajo costo computacional. Se efectuaron pruebas de este algoritmo en otras aplicaciones como detección de fallos en la producción de válvulas (de admisión y escape) y pistones de motores de explosión con muy buenos resultados. Adicionalmente en muchas aplicaciones industriales es fácil conocer la posición exacta del objeto a examinar, evitándose así la primera etapa de detección del centro.

Para muchas aplicaciones es útil realizar binarizaciones de la imagen con distintos niveles de umbral para detectar diferentes errores de fabricación. Esto no agrega complejidad al algoritmo ya que en una única pasada se puede recoger esta información, como lo sugiere el pseudocódigo de la figura 13 y el ejemplo de la figura 14.

```
for (c/punto en la imagen)
      { sectorPunto = calculoSectorPertenencia(punto);
        if (UMBRAL_1i < punto < UMBRAL_1s) sectores_1[sectorPunto]++;
        if (UMBRAL_2i < punto < UMBRAL_2s) sectores_2[sectorPunto]++;
        if (UMBRAL_3i < punto < UMBRAL_3s) sectores_3[sectorPunto]++;
            ...
        if (UMBRAL_ni < punto < UMBRAL_ns) sectores_n[sectorPunto]++;
      }
```

Figura 13. Pseudo código para la detección de fallas en piezas con simetría de revolución

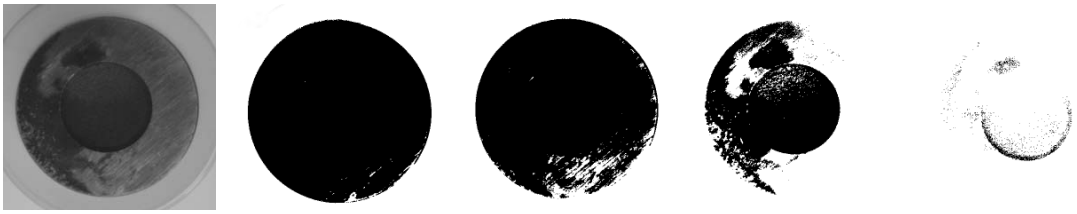


Figura 14. Pieza cilíndrica de acero mecanizada con defectos y las diferentes binarizaciones

Referencias

[1] R.C. Gonzalez. Digital Image Processing. Addison-Wesley. 1991.
[2] B.K.P. Horn. Robot Vision. The MIT Press. 1986.
[3] R. Jain, R. Kasturi and B. Schunck; Machine Vision, McGrau-Hill 1995.
[4] J. G. Jiménez. Visión por computador, Paraninfo 2000.
[5] J.R. Parker. Grey level thresholding in badly illuminated images, IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol.13. 1991.
[6] J.R. Parker. Algorithms for image processing and computer vision. John Wiley&Sons Inc. 1995
[7] N. Otsu. A threshold selection method from Grey-Level Histograms, IEEE Transactions on Systems, Man and Cybernetics. Vol. 9. 1:377-393. 1979.