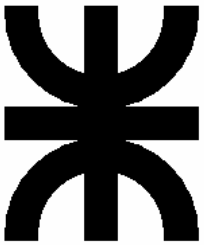


UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL SAN NICOLAS

INGENIERÍA EN ELECTRÓNICA

PROBLEMA DE INGENIERÍA

TÉCNICAS DIGITALES III

DETECCIÓN DE PLIEGUES EN BOBINAS

Integrantes**:**

- Girotti Leonardo

Docentes**:**

- Profesor: Poblete Felipe
- Auxiliar: Gonzalez Mariano

AÑO 2008

INDICE

OBJETIVOS DEL TRABAJO

Esta aplicación se desarrollo con el fin de detectar posibles pliegues en las bobinas que puedan causar una ruptura de esta.

Los pliegues son formados en el laminador debido a oscilaciones de la chapa al momento de ser "pisada por los rodillos" y estos son causantes de cortes en etapas posteriores.

Una ruptura en la sección de estañado generaría una demora importante y una perdida de material.

MATERIAS INTEGRADAS

• Informática 2.
• Física 3.

POSIBLES APLICACIONES

• Detección de pliegues superficiales en chapas y otros materiales no reflejantes.
• Desarrollo y estudio de funciones y librerías de visión artificial.

PROFESORES ENTREVISTADOS

• González Mariano (Estructuras y uniones en C++, sockets para la comunicación tcp/ip)

BIBLIOGRAFÍA

• *Sitios de Internet.*
    • Blogs acerca de OpenCV en yahoo.com.
    • Papers acerca de iluminación y ejemplos de filtros.

• Manual de Borland Builder 6; Programación en C. Byron S. Gottfried; Documentación incluida en el paquete le librerías de OpenCV.

DESARROLLO

INTRODUCCIÓN

Los pliegues son uno de los causantes de cortes en las bobinas, estos generan problemas debido a demoras y pérdidas de material.

Los pliegues se generan debido a que los rodillos al laminar, ante determinadas circunstancias, pisan y solapan la chapa.

Estos debido a la potencia del laminador quedan ocultos y son difíciles de detectar.

Se plantea como solución a este asunto la detección óptica de los pliegues aprovechando las ventajas que proporcionan los métodos de iluminación que se estudiarán más adelante.

Se utilizará una cámara lineal que es óptima para este tipo de aplicación, esta se sincroniza con la línea para mantener en todo momento la proporción de la relación píxel – mm.

La cámara será montada sobre la línea de recorte lateral. Esta arma imágenes a partir de la toma de 300 líneas sincronizada con el avance de la bobina y la envía por camera link.

Las imágenes son analizadas mediante un software dedicado. En caso de pliegue se genera una alarma, se salvan las imágenes correspondientes y se almacenan los datos en un historial.

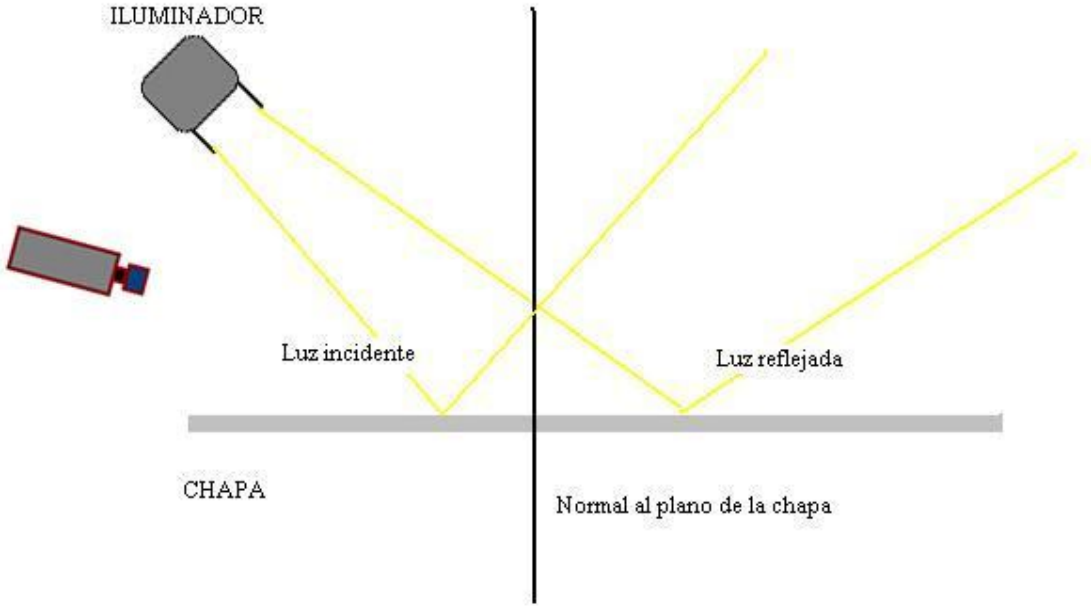
TEORÍA DE FUNCIONAMIENTO

Iluminación.

Se explicará la teoría de iluminación aplicada con el efecto de resaltar el defecto que se quiere detectar.

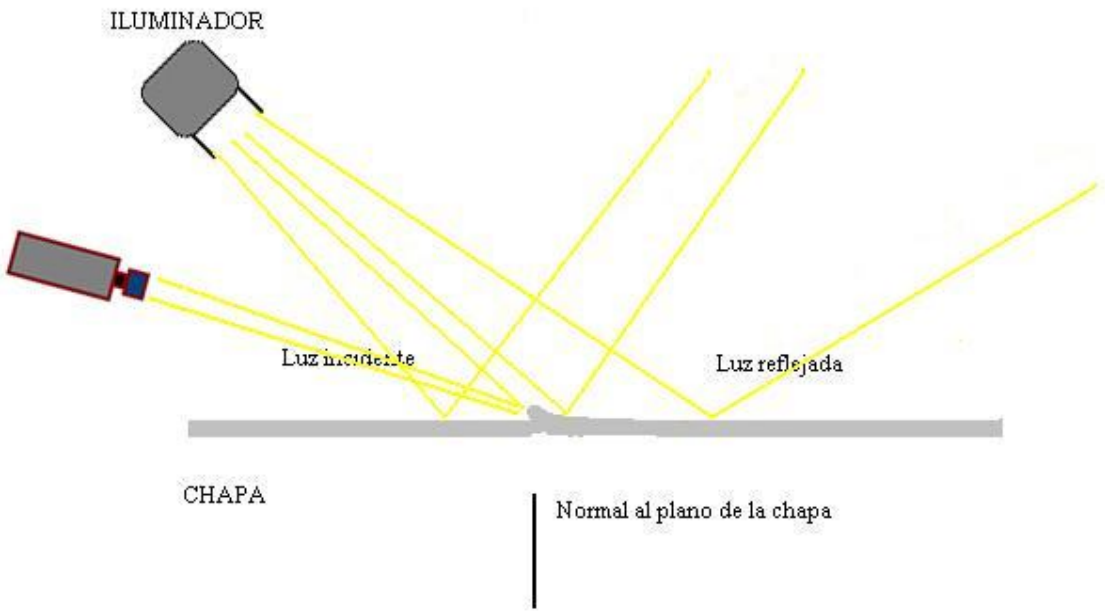
Existen múltiples métodos de iluminación que resaltan características distintas de una superficie. Para este caso se aplicó la visión del campo oscuro.

Chapa en buen estado:



Como se observa en el gráfico la luz es reflejada con un ángulo igual (con respecto a la normal) al de incidencia. Esto hace que se vea la chapa iluminada pero no el reflejo directo de la luz, es por eso que se llama campo oscuro.

Chapa con pliegue:



Ante un pliegue se cambia el ángulo de reflexión provocando una línea brillante que se destaca.

Para la iluminación se utilizó dos reflectores alójenos de 500 W con pantallas para encausar la luz, dispuestos a igual distancia del centro de la chapa a 45 ° de esta.

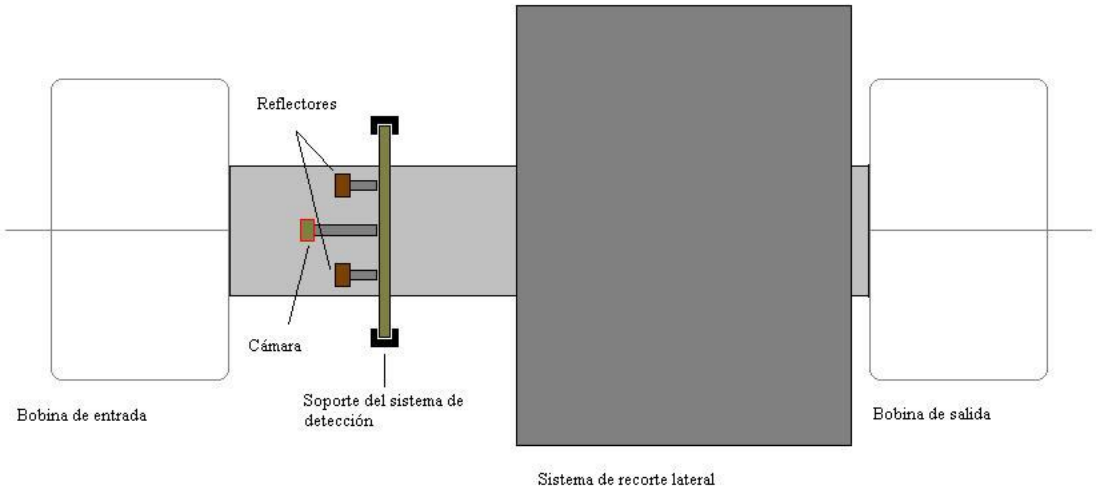
Para este tipo de aplicaciones es mejor utilizar luz estructurada para lo que vienen iluminadores que utilizan paneles de leds. Esto fue probado pero por las características de la zona la potencia del iluminador no fue suficiente. El mayor inconveniente es que por encima de la línea pasa el punte grúa y los reflectores instalados en esta son muy potentes y saturan la imagen.

Montaje.

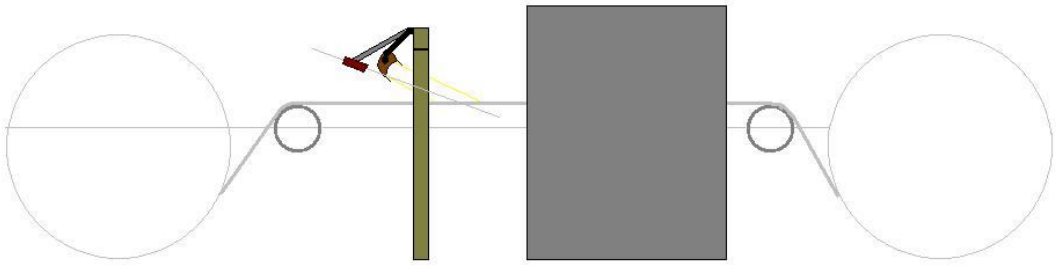
El sistema se montará antes del equipo de recorte lateral.

A continuación se muestra un esquema básico del montaje.

Esquema visto de arriba:



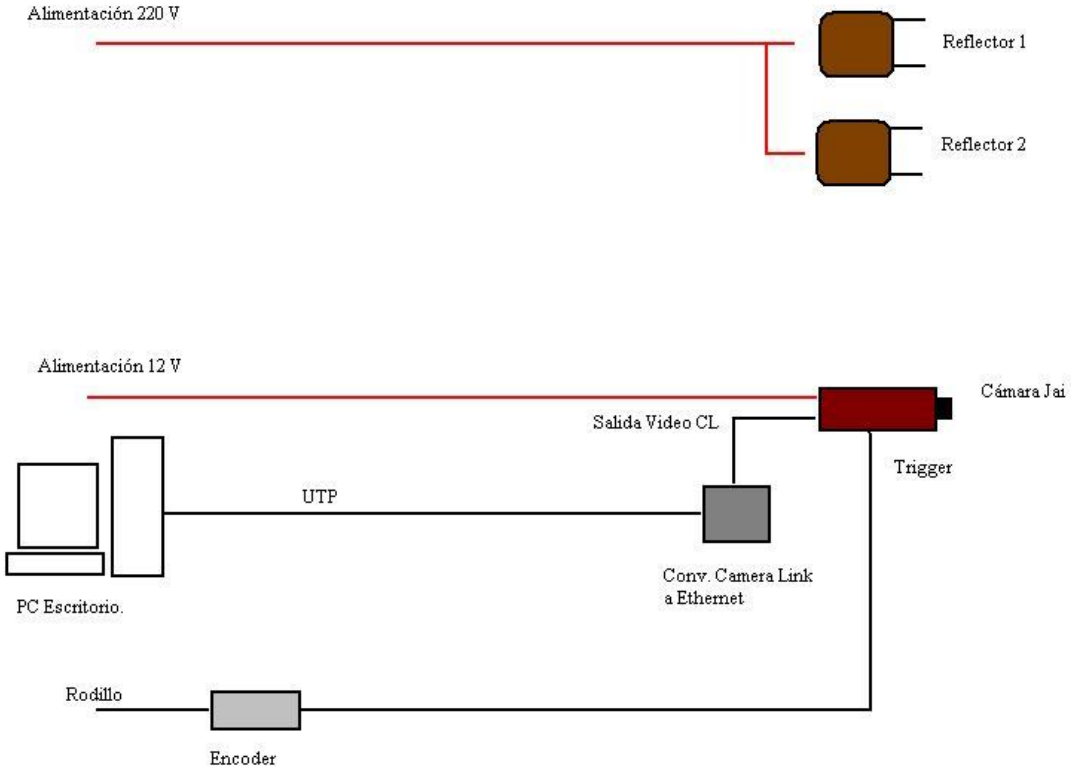
Esquema Lateral:



Esquema de conexión del Hardware.

A continuación se observa el diagrama de interconexión entre los dispositivos de hardware que componen el sistema.

Se utilizó un encoder que proporciona el trigger a la cámara a partir del avance del rodillo. Este tiene una relación tal que la cámara dispara una vez por cada milímetro de avance de la chapa.



Funcionamiento del sistema y elección de los dispositivos.

Cámara utilizada.

La cámara utilizada es una lineal de resolución 2 MP con ccd de 1". Se eligió esta debido a que es ideal para la detección de defectos superficiales ya que concentra toda su resolución en cada línea permitiendo imágenes de alta definición con sólo 2 MP. Esta se configura para que envíe imágenes de 300 líneas.

Su ubicación es debajo de los reflectores apuntando a la línea de iluminación.

La velocidad de adquisición máxima es 18 K. El tiempo de adquisición debe ser calculado para permitir la frecuencia necesaria.

Lente.

Las bobinas que se observaran tienen un ancho máximo de 1650 mm. Se elijió una lente de tipo Fujinon de 1" con una longitud focal de 12,5 X. A continuación se muestran los cálculos aproximados que demuestran las distancias elegidas.

Datos: Ancho bobina: 1650.
Ancho de observación: 2000
Long. Focal lente: 12,5 X.
Distancia entre Cámara y bobina: 1000
CCD: 25.4 mm

Long. Focal lente = CCD * Distancia / Ancho de observación = 12,7

De los datos y el cálculo realizado elegimos la lente de 12,5 X y 1 ".

Resolución vertical y elección del encoder.

La velocidad de la línea va desde 0 hasta 600 m/min (10.000 mm/seg).

La resolución horizontal se calcula a partir del ancho observado:

Res Horiz. = ancho obs. / cant. Píxeles ccd = 2000 / 2048 ≈ 1 mm/pix.

Por esto elegiremos el encoder de manera tal que la resolución vertical sea de 1 mm/pix, esto es uno que envíe un pulso por cada milímetro de avance. Como este será montado al primer rodillo de la tijera, el cálculo es:

Cant. de pulsos por vuelta = Perímetro de rodillo en mm.

La cámara será disparada cada 1/10.000 s. El tiempo de exposición máximo posible será 0.1 ms lo que es bastante poco y por lo tanto será necesario muy buena iluminación de la superficie a controlar. Es por esto que se utilizarán reflectores en lugar del iluminador de leds.

Obtención de imágenes.

Una vez que se realizaron los cálculos necesarios y se eligieron los dispositivos apropiados el siguiente paso fue la realización del montaje.

Una vez terminada la instalación se tomaron muestras con el fin de empezar a estudiar el problema.

Se analizaron distintas imágenes. Fue necesario apartar aquellas bobinas con defectos para poder capturar ejemplos de pliegues ya que estos son muy escasos y por lo tanto es muy difícil obtener muestras.

En este tipo de aplicaciones es fundamental la cantidad de muestras para el diseño del software.

Diseño del Software.

Consideraciones previas.

A velocidad máxima de línea la cámara será disparada aproximadamente 10.000 veces por segundo. Esto implica que se generarán aproximadamente 30 imágenes por segundo.

Cuando diseñemos nuestro proceso tendremos que considerar que estaremos limitados en el tiempo de inspección a razón de 1/30 s.

El tiempo de inspección depende de los algoritmos utilizados y de la PC sobre la que se trabaje.

Análisis de la imagen.

Si bien hay muchos métodos y procedimientos para las aplicaciones de procesamiento de imagen, cada nuevo proyecto es diferente a los demás.

Al trabajar con imágenes entran en juego un gran abanico de factores que hacen imposible la reutilización de los procedimientos entre aplicaciones similares.

Uno de los pasos más importantes es el de la obtención de muestras, tanto de aquellas que no tienen defectos como de las que sí.

Debido a que el método utilizado consiste en resaltar los defectos mediante la iluminación una de las principales consideraciones a tener en cuenta es que el defecto tendrá un alto contraste con el resto de la bobina.

Otro factor importante para el filtrado de ruido es la forma. Los pliegues tienen una forma alargada, con un ángulo entre 60 y 85 grados para aquellos que se orientan de derecha a izquierda y entre 95 y 120 para los orientados en sentido contrario.

Se debe considerar también que estos defectos son muy angostos y por lo tanto su área no es muy grande.

- Filtros.

A continuación se numeran los filtros utilizados según su orden y se provee una explicación acerca de su función.

Sobel.

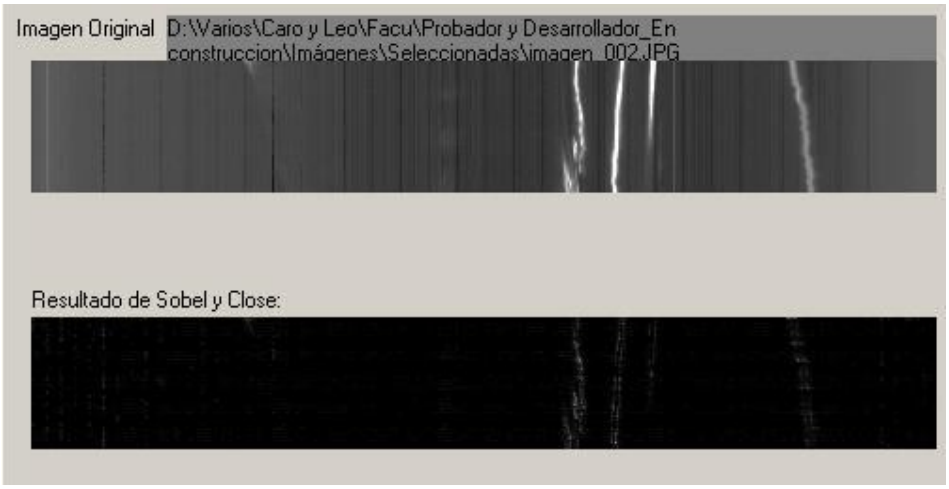
Este es una operación morfológica, o sea que actúa directamente sobre la morfología de la imagen, que realiza un gradiente de los píxeles en una dirección determinada.

En nuestro caso se realiza el gradiente en Y, esto provoca que se resalte aquellas sucesiones que tengan incremento o decremento en la intensidad de los píxeles y elimina aquellos baches en los que la intensidad se mantiene constante.

Este se utiliza para eliminar las hebras delgadas que aparecen en la imagen reduciéndolas a una simple línea y aquellos manchones provocados por ruidos de iluminación (paso de la grúa).

Esta herramienta es fundamental a la hora de resaltar los defectos buscados debido a su orientación y la textura en la iluminación en estos.

A continuación se observa la imagen adquirida y el resultado de la operación.



En la imagen se ve también el efecto de la operación close que será explicada más adelante. Puede verse que se redujo al veteado de la imagen.

Close.

Esta operación es la combinación de dos operaciones. Primero se realiza un “dilate” y luego un “close”. La primera rellena los puntos negros que se encuentran en medio de puntos con alta intensidad tomando los máximos en torno a cada punto. La segunda hace el procedimiento inverso.

La operación close genera que se rellenen los puntos oscuros en medio de puntos de alta intensidad y además elimina aquellos puntos luminosos que estén muy separados del resto. Con esto logramos marcar los flancos luminosos y resaltar contrastes.

El resultado se observa conjunto a el resultado de sobel en la imagen anterior.

Mascara.

De observar la imagen anterior deducimos que se ha eliminado en buena parte los ruidos. También se ve que se redujo bastante la intensidad de los pliegues.

Se utiliza la herramienta de máscara con la imagen modificada como filtro generando así que se refuercen los píxeles que ya estaban.

A continuación se ve el resultado.

Threshold.

Este filtro binariza la imagen llevando a todo píxel que esté por arriba del umbral a la saturación y a todos los que estén debajo a cero.

Este se utiliza para que una vez que se halla resaltado aquello que se desea pueda eliminarse el resto.

Además la detección de bordes, formas y etc se hacen sobre imágenes binarizadas.

Logramos, mediante los filtros vistos hasta ahora, resaltar considerablemente los defectos deseados. Es posible ahora utilizar el threshold para eliminar los grises.

Vemos a continuación la binarización de la imagen anterior.

En este paso ya hemos eliminado la mayor parte del ruido y logramos rescatar aquellos blobs que nos interesan.

Filtro por tamaño de blobs.

Existen casos en que, por determinadas circunstancias, no es posible eliminar blobs generados por ruido que no reflejan un defecto real.

Uno de estos casos es el ruido generado por las grúas, que cuentan con una potente iluminación. El reflejo generado sobre la bobina es visto como grandes blobs que confunden a los discriminadores. Para este caso se diseño este filtro. Este encuentra todos los blobs y saca el área

correspondiente, a partir de esto se puede eliminar aquellos defectos encontrados que cuenten con un área mayor a un umbral determinado.

Vemos a continuación el resultado de threshold para un caso de ruido por grúa y el resultado del filtro por tamaño de blob.

Transformada de Hough.

Este es un método de detección de líneas. Recorre la imagen binarizada y genera rectas que contienen una determinada cantidad de píxeles saturados. La cantidad de píxeles es ajustable junto con otros parámetros.

La idea es aprovechar la forma de los defectos, alargados y angostos con inclinación diagonal, para encontrarlos por el ángulo que forman con la normal.

Este método nos permite eliminar aquellas líneas verticales y horizontales que todavía quedan en forma de ruido.

Después de utilizar esta herramienta se genera un lazo que discrimina a las líneas encontradas según su ángulo.

A continuación se observa el resultado del análisis ante un caso de pliegues múltiples.

Se observa que los pliegues fueron encontrados.

Tenemos el caso antes visto del ruido por la grúa:

En este caso se logra eliminar prácticamente todo el ruido y el resto se discrimina por tener ángulos de orientación fuera del rango buscado.

RESULTADOS DE LAS PRUEBAS

Funciones utilizadas en el desarrollo del algoritmo de detección.

Las funciones utilizadas para realizar la detección de pliegues que fueron presentadas son el resultado de variadas pruebas.

Fueron utilizadas distintas combinaciones de algoritmos para analizar los sets de muestras. De estas pruebas se obtuvo el procedimiento final que es el presentado.

En este anexo se presentaran algunos de los filtros básicos que pueden ser usados para alterar la imagen primaria.

Smooth.

O suavizar como figura en la pantalla. Se realiza una convolución de la imagen contra una matriz que promedia cada píxel con los valores de los píxeles cercanos.

El tamaño de la matriz puede variar, mientras más grande más dispersión de la imagen.

Este filtro se utiliza para eliminar el efecto sal y pimienta.

Threshold.

Este simplemente compara los valores de cada píxel asignándole máxima intensidad a los que superen un umbral elegido y nulo a los que no.

Convolución.

Esta opción junto con la opción Matriz permite que le apliquemos a la imagen una convolución con la matriz que diseñemos. Esto permite que realicemos derivadas, promedios, y operaciones morfológicas como queramos.

Máscara.

Es simplemente realizar una operación lógica AND entre dos matrices. Se utiliza para obtener una imagen con los valores originales de píxeles después de varias operaciones para filtrar partes no deseadas.

Dilate.

Es una operación morfológica simple. Por cada píxel toma el valor del vecino con mayor intensidad. Permite llenar huecos en los contornos.

Erode.

Similar a la anterior sólo que esta toma el valor de menor intensidad. Permite llevar una figura a su esqueleto.

Sobel.

Es una derivada de la imagen en la dirección que elijamos (X ó Y). Utiliza la convolución contra un tipo de matriz especial. Se utiliza para resaltar bordes eliminando el relleno.

Grad.

Realiza un gradiente. Utilizado para detectar progresiones de intensidad.

Contorno.

Existen muchas algoritmos y combinaciones para reasaltar contornos:

- Por borde. Se utilizan derivadas para resaltar los bordes de distinto ángulo y se suman los resultados.
- Por dilatación y erogación. Se resta el resultado.
- Por conectividad de píxeles.

CONCLUSIONES

Como conclusión y en forma de ayuda para aquél que quiera ingresar al mundo de la visión artificial puedo decir que existen numerosos métodos para resolver las distintas situaciones que queramos.

Para cada caso en particular se puede optar por distintos tipos de iluminación acorde con el objetivo, como se explico más arriba, que conllevarán cada uno a distintas posibilidades en el desarrollo del código a utilizar.

Cuando queramos encarar un problema deberemos informarnos muy bien de todas las posibilidades ya estudiadas para nuestro caso e inclusive analizar otras. Luego cada problema es único debido a las infinitas variables que se presentan (lugar, luz de sol, suciedad, presencia de partículas, etc) por lo que existirá un trabajo de prueba y error inevitable por parte del desarrollador.

ANEXOS:

LISTADOS DE PROGRAMAS

Programa Principal.

Las librerías utilizadas para la inspección de imágenes son la versión 1 de las opencv. Estas fueron compiladas e importadas para poder utilizarlas desde borland. Estas son muy potentes y además son FREE. Otra de sus ventajas es que cuentan con un gran soporte en blog de yahoo.

A continuación se muestra el código utilizado para el análisis de las imágenes.

El resto del código utilizado en el programa puede ser consultado directamente en las fuentes. Todo el código está debidamente comentado para un fácil seguimiento del software.

```
//--------------------------------------------------------------------------
--------------------
//----------------------------------- Análisis principal ---------------------
-----------------
//--------------------------------------------------------------------------
--------------------

void analisisprincipal(void)
{
  float* line;
  double angulo=0;
  int x=0;
  int y=0;
  int angh=0;
  String ang=" ";
  bool alarma=false;    //bandera de alarma por pliegues.
  char nomimg[200];     // almacenamiento de imágenes.
  FILE *fp_log;         // almacenamiento del historial de pliegues.

// Variables para el filtro por área de blob.
  CvSeq* contour = 0;
  CvMemStorage* storage = cvCreateMemStorage(0);        // donde se almacenan
los contornos obtenidos
```

```
  double area=0;
  String areastring=" ";

// Variables para la transformada de hough.
  CvMemStorage* storage_h;
  CvSeq* lines = 0;
  LARGE_INTEGER tiempoini, tiempofin, frec;
  float tiempoproc=0;
  double frecuencia=0;

  QueryPerformanceCounter( &tiempoini );        // Comienzo a contar para
obtener el tiempo de análisis.

//-------------------------------- Comienzo análisis principal de la imagen ---
------------

  Form1->Label40->Color=clSkyBlue;      // reseteo el color de la bandera de
falla.

// Filtro Sobel.

  cvSobel(ImagenOriginal ,Imagen16 ,0 ,1 ,3 ); // Se hace en sentido vertical.

  cvConvertScaleAbs( Imagen16, ImagenModificada2, 1, 0 );

// Filtro Close.

  cvMorphologyEx( ImagenModificada2, ImagenModificada2, ImagenBuffer, kernel, 3,
1 );

// Realizo una máscara para intensificar aquellos píxeles que hallan pasado los
filtros.

  cvZero( ImagenBuffer );
  // copy edge points
  cvCopy( ImagenOriginal,ImagenBuffer, ImagenModificada2 );      // copia de
imagenoriginal a imagenbuffer con imagenmodificada como máscara
  ImagenModificada2 = cvCloneImage( ImagenBuffer );              // salva el
resultado en imagen modificada

// aplico threshold para que queden los blobs
```

```
  cvThreshold( ImagenModificada2, ImagenModificada2, 135,255,0);

//--------------------------------------------------------------------------
---------------------
// Bloque de filtrado por área de blob. Este filtro calcula el área de los blobs
presentes en // la imagen y elimina aquellos que sobrepasen un determinado
límite.
//--------------------------------------------------------------------------
---------------------

// Encuentro los contornos de la imagen.

  cvFindContours( ImagenModificada2, storage, &contour, sizeof(CvContour),
CV_RETR_EXTERNAL, CV_CHAIN_APPROX_SIMPLE );

  cvZero( ImagenModificadaColor );

  CvScalar color;
  color.val[0]=0;

// Recorro los contornos encontrados y elimino de la imagen aquellos con un área
mayor a lo especificado.

  for( ; contour != 0; contour = contour->h_next )
  {
    area=-cvContourArea( contour, CV_WHOLE_SEQ );       // Calcula el área de
cada contorno
    if(area>26500)
      cvDrawContours( ImagenModificada2, contour, color, color, 2, CV_FILLED, 8
);
  }

//--------------------------------------------------------------------------
---------------------
// Aplico la transformada de hough para obtener la orientación de las
principales rectas formadas por los pixeles.

  storage = cvCreateMemStorage(0); // crea un deposito para las coordenadas de
las líneas devueltas.
```

```
  cvCvtColor( ImagenModificada2, ImagenModificadaColor, CV_GRAY2BGR );
  ImagenBuffer = cvCloneImage( ImagenModificada2 );      // El método
probabilístico altera la imagen.

  lines = cvHoughLines2( ImagenBuffer, storage, CV_HOUGH_PROBABILISTIC, 1,
CV_PI/180, 50, 40, 5 );

// Recorro las líneas encontradas calculando es ángulo para cada una y
verificando que este dentro del rango requerido.

  for( i = 0; i < lines->total; i++ )
  {
    CvPoint* line = (CvPoint*)cvGetSeqElem(lines,i);

    y=line[0].y - line[1].y;    // Longitud del cateto opuesto.
    x=line[0].x - line[1].x;    // Longitud del cateto adyacente.

    if(x==0)
      angulo=90;
    else
      angulo=atan(y/x)*(180/CV_PI)*(-1);  // para que me de en ángulos, la
correccion es por el eje de coordenadas de la imagen

    if(angulo<0)        // En caso de que los ángulos den negativos los pasa a
su complementario.
      angulo=angulo+180;

// Controlo el ángulo de las tendencias encontradas. si se dan las condiciones
marco como pliegue. En caso d detectar algún defecto pongo la bandera de alarma
en trae.

    if(((30<angulo)&&(angulo<88))||((92<angulo)&&(angulo<150)))
    {
      cvLine( ImagenModificadaColor, line[0], line[1], CV_RGB(255,0,0), 3, 8 );
      alarma=true;      // Si se encuentra un pliegue se activa la alarma.
    }

  }     // fin del FOR.

// Si se levantó la bandera de alarma se activa la bandera de pantalla,se logea
el evento y se salva la imagen.
```

```
  if(alarma)
  {
    Form1->Label40->Color=clRed;      // Si hay pliegue pongo la alrma en rojo.
    Form1->contador->Caption=(contadorfallas++);

    sprintf(nomimg,"Pliegue_%03d.jpg",contadorfallas);
    cvSaveImage( nomimg, ImagenOriginal );

// Guardo los datos de tiempo, N° de pliegue e imagen en el historial.
    fp_log = fopen( "Bobina_001.txt", "a");
    if( fp_log != NULL )
    {
      fprintf(fp_log,"%s : Bobina N°: 001,      Pliegue N°:%d,          Imagen:
Pliegue_%03d.\n", TDateTime::CurrentDateTime().FormatString("HH:mm:ss").c_str(),
contadorfallas, contadorfallas);
      fclose(fp_log);
    }

    alarma=false;
  }

  QueryPerformanceCounter( &tiempofin );

  QueryPerformanceFrequency( &frec );
  frecuencia = (double)(frec.QuadPart) / (double)1000;

  tiempoproc = (float)(tiempofin.QuadPart-tiempoini.QuadPart)/frecuencia;

  Form1->tciclo->Caption= FormatFloat("0.0",tiempoproc );

  Form1->cvshowBMPresultado(ImagenModificadaColor);
}
```

Pantalla principal.

A continuación se observa un print screen de la pantalla principal del software.

La apariencia, como se observa, es bastante simple. El análisis se inicia con el botón iniciar y se detiene con detener.

Una vez comenzado el análisis el software no necesita de ningún operador. Este se encarga de analizar cada bobina e informar ante un evento de pliegue, además almacena las imágenes correspondientes a los pliegues encontrados por lo que no es necesario estar pendiente del software.

Se indica en pantalla la cantidad de defectos encontrados para cada bobina y además el tiempo que le toma al software en realizar el proceso de obtención de imagen y análisis.

La idea es que al terminar el pasaje de la bobina por la línea, el operador controle la cantidad de fallas encontradas y si existe alguna, alerte que la bobina tiene probabilidades de cortarse en etapas posteriores. En caso de duda las imágenes de los pliegues están disponibles para poder estudiarlas. Además se tiene el logeo de los sucesos.

El botón configuración es sólo para desarrolladores. Este permite el análisis de las imágenes para adaptar el software ante algún cambio en las condiciones o para modificar el tipo de defectos que se busca.

Este software es bastante dinámico y la parte de configuración permite jugar con las diferentes herramientas que provee las librerías opencv de manera de poder hacer pruebas de rápida inspección para resolución de diferentes situaciones.

Se observa a continuación la pantalla correspondiente a configuración.