Práctica 6
Análisis mediante técnicas morfológicas

- Detección de defectos en placas de circuito impreso

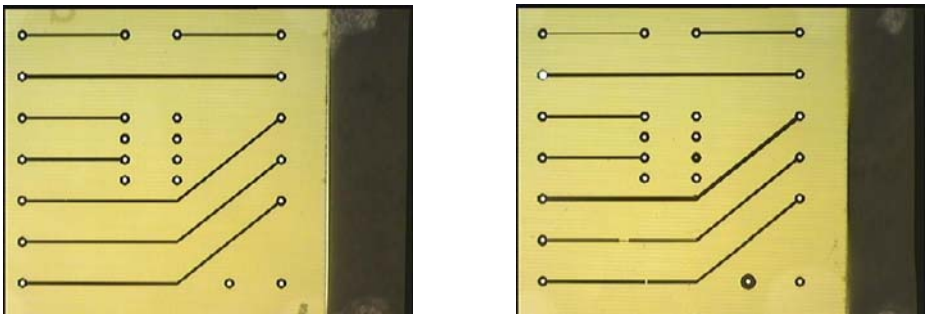
Robótica y Visión por Computador
Ing. Telecomunicaciones

Universidad Miguel Hernández

Objetivo de la práctica:

El objetivo de la práctica es profundizar en las técnicas morfológicas de procesamiento de imágenes utilizando las librerías OpenCv.

Las imágenes de trabajo son dos imágenes a color reales. Cada una de ellas se corresponde con un circuito distinto. Las imágenes tienen un tamaño de 640x480 pixeles y se encuentran disponibles en la web de la asignatura.



Imágenes reales de dos placas de circuito impreso

En ellas se distinguen tres tipos de objetos:

-Conductores, de tonalidad más oscura.
-Placa, de tonalidad intermedia.
-Taladros, de tonalidad más clara.

Se supone conocidos, aproximadamente, los siguientes datos:

-Rango de anchura de los conductores: 3-6 píxeles (ambos inclusive).
-Rango de tamaño de los radios de los taladros: 2-3 píxeles (ambos inclusive).
-Rango de tamaño de los radios de los Pad: 6-9 píxeles (ambos inclusive).

La práctica se debe realizar para cada una de las imágenes.

El desarrollo de la práctica se divide en dos partes: la primera en la que se adaptan las imágenes de prueba, y la siguiente en la que se detectan por separado las características de los taladros y de los conductores. Se aconseja seguir los siguientes pasos para la realización de la misma:

Realización de la práctica:

1. En primer lugar se debe abrir el workspace creado en la práctica 0 o bien generar uno nuevo tal y como se explicó en dicha práctica.
2. Añadimos un nuevo proyecto al workspace, que llamaremos **practica6**
3. Añadimos un fichero fuente vacío que llamaremos practica6.c, y añadimos las librerías necesarias para trabajar con las OpenCv.

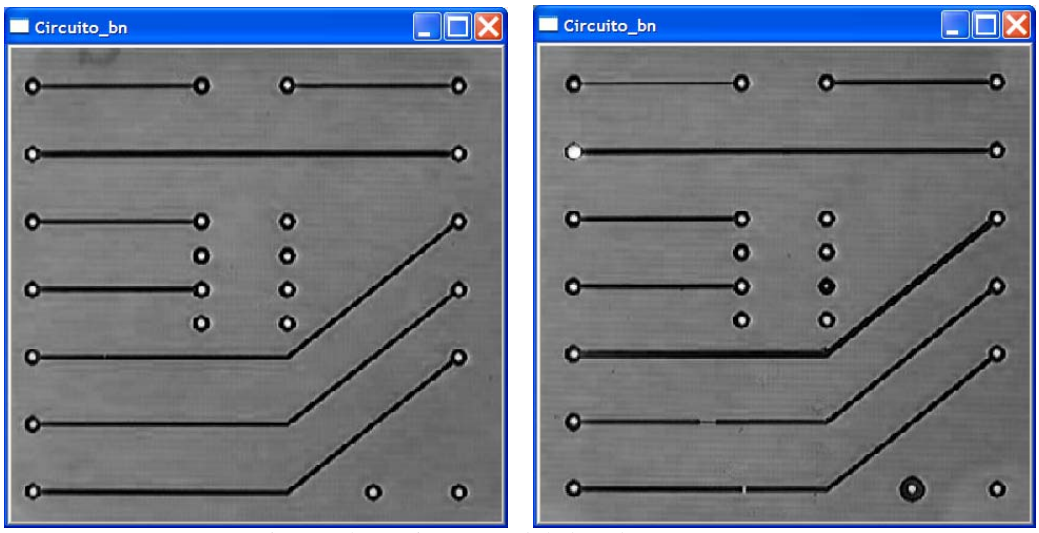
4. La primera etapa de la práctica va a consistir en adaptar las imágenes reales, de forma que podamos trabajar con ellas mediante técnicas morfológicas. Para ello vamos a seguir los siguientes pasos:
   a. Después de cargar la imagen en color,

```
src1 = cvLoadImage( name1, 3 );
```

      se va a obtener una imagen en escala de grises que contenga la misma información que la inicial. Una posible forma de realizar esta acción es dividir los canales de la imagen en color y quedarnos con uno de ellos. Para ello se utiliza el **COI** (Channel of Interest) y la función `CvCopy` (de esta forma sólo se copia en la nueva imagen el canal establecido). En este caso se va a elegir como COI el canal azul (channel 1 en OpenCV).
   b. A la vez que se realiza el paso anterior se utiliza tambien **ROI** (Region of Interest) para definir la zona de análisis, de forma que eliminamos el borde exterior capturado en la imagen real. En este caso se define un rectángulo de dimensiones 465x450. Este rectángulo define por tanto el tamaño del resto de imágenes que se utilizan en la práctica.

```
CvRect rect = { 10, 10, 465, 450 }; // los parametros son xini, yini, width, height
CvSize imgSize; // tamanyo de la imagen
imgSize.width = 465; // el tamaño de la imagen de trabajo
imgSize.height = 450; // es de 465x450 pixels
cvSetImageCOI(src1,1);              //setImageCOI con 1 se elige el canal azul (BGR)
cvSetImageROI(src1,rect);           //para coger el ROI
cvCopy(src1,circuito_bn,NULL);
```

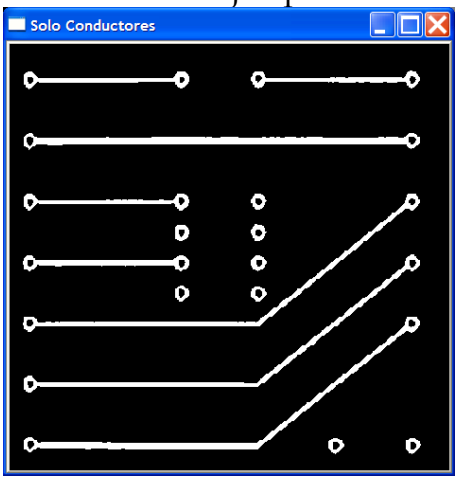
      De esta forma se obtienen las imágenes **`circuito_bn`** que se muestran a continuación:

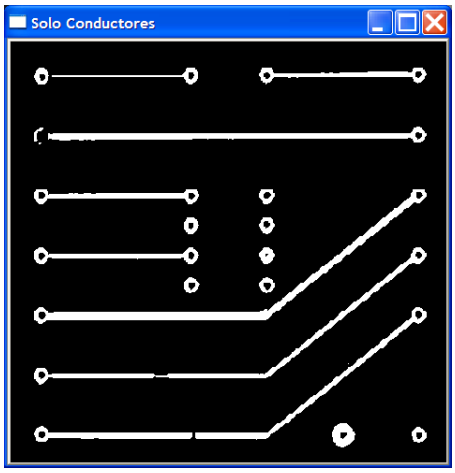


Imágenes de un único canal de las placas impresas

   c. Se utilizan las técnicas de closing y opening unidas a la umbralización para obtener las siguientes imágenes intermedias:

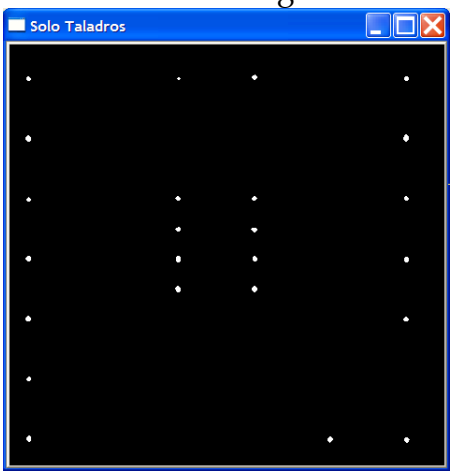
- Imagen **`solo_cond`**. Se trata de una imagen binaria que contiene sólo los conductores. Para obtenerla se realiza una operación de closing seguida de un opening utilizando para ello un elemento estructurante 2x2 en forma de cruz. Posteriormente se realiza una umbralización con valor 85 y se invierte la imagen, de forma que se obtienen las siguientes imágenes. El código ejemplo se muestra a continuación:

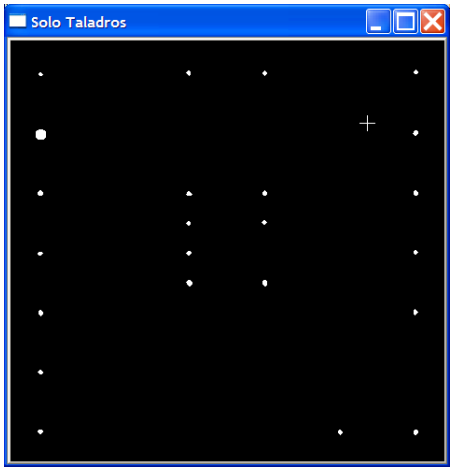




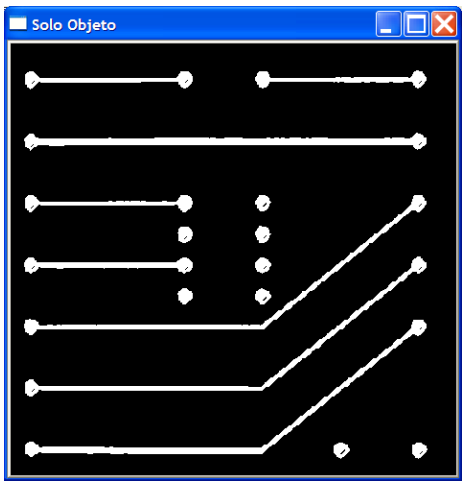

```
IplConvKernel* element = 0;
element = cvCreateStructuringElementEx(2,2,1,1,CV_SHAPE_CROSS,0);
cvDilate(circuito_bn,temp,element,1);       //closing= dilate+erode
cvErode(temp,dest,element,1);        //
cvErode(dest,temp,element,1);        //opening= erode+dilate
cvDilate(temp,dest,element,1);
cvReleaseStructuringElement(&element);
cvThreshold(dest,solo_cond,85,255,CV_THRESH_BINARY_INV);
```

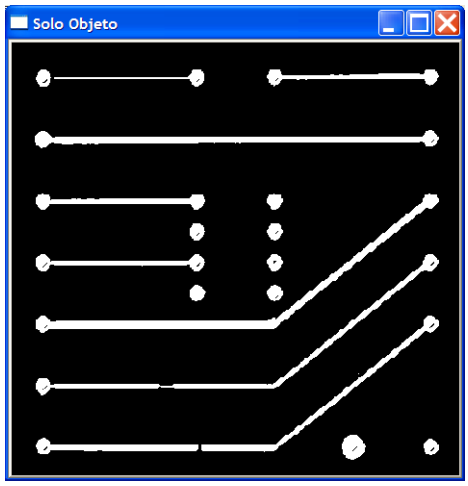
- Imagen **`solo_taladros`**. Se trata de una imagen binaria que contiene sólo los taladros. Para obtenerla se realiza una operación de closing seguida de un opening utilizando para ello un elemento estructurante 3x3 en forma de cruz. Posteriormente se realiza una umbralización con valor 170, de forma que se obtienen las siguientes imágenes:





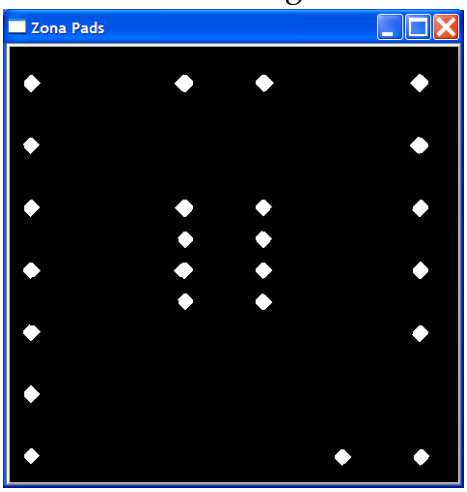
- Imagen **`solo_objetos`**. Se trata de una imagen binaria que define la zona conjunta de los conectores y los taladros. Para obtenerla se debe realizar una dilatación de la imagen **`solo_taladros`** con 3 iteraciones con un elemento estructurante 3x3 en forma de cruz y posteriormente realizar una operación `OR` con la imagen **`solo_cond`**. El código y las imágenes se muestran a continuación.

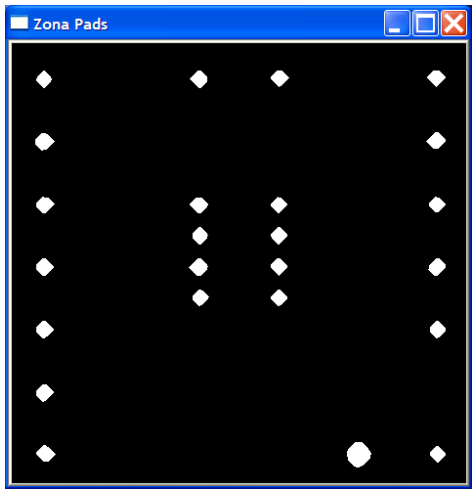





```
element = cvCreateStructuringElementEx(3,3,1,1,CV_SHAPE_CROSS,0);
cvDilate(solo_taladros,temp,element,3);
cvOr(temp,solo_cond,solo_objeto,NULL);
cvReleaseStructuringElement(&element);
```

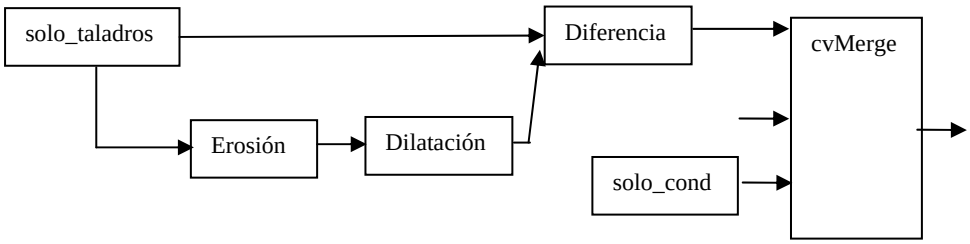
- Imagen **`zona_pads`**. Para hallar la zona de influencia de los Pads se dilata dos veces, con un elemento estructurante en cruz, la imagen **`solo_objeto`** (para evitar posibles zonas no homogéneas) y a continuación se erosiona ocho veces la imagen, con un elemento estructurante en cruz (según tamaño máximo posible de los conductores), para posteriormente dilatar seis veces (ocho menos dos) la imagen resultante. Se deben obtener las siguientes figuras:



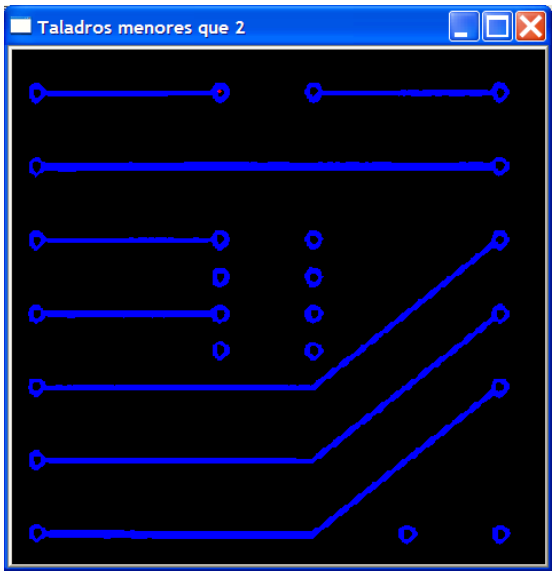


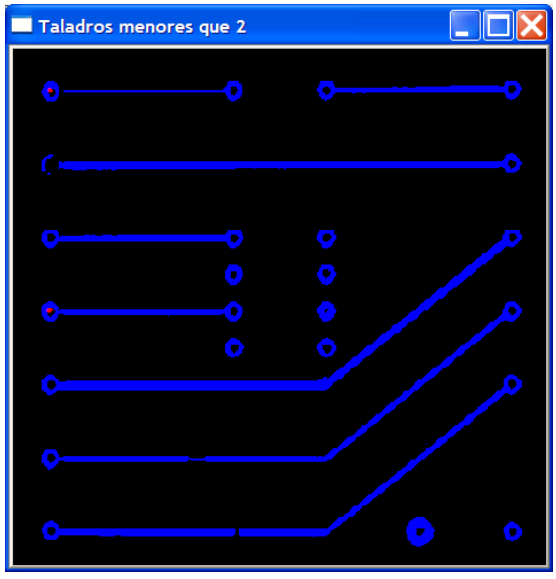
5. Una vez obtenidas las anteriores imágenes, se va a proceder al análisis de las imágenes y a la detección de errores. Para tener una mejor visualización de los defectos, se van a utilizar imágenes en color, de forma que el defecto se coloque en un canal y el fondo en otro canal (Para ello se utiliza la función `cvMerge` de la forma mostrada en el código). En esta memoria se muestra como ejemplos cómo identificar los taladros menores que 2 pixeles de diámetro y cómo identificar los pads que no poseen taladro, mientras que se pide escribir el código para detectar los taladros de diámetro mayor que 3 pixeles y la determinación de las pistas con un grosor menor de 2 pixeles.

    a. Determinación de taladros menores de 2 pixeles. Se muestra el algoritmo utilizado y el código que lo implementa en OpenCv. En el bloque "erosión" se realizan 2 iteraciones con el fin de eliminar los taladros con un radio inferior a 2 pixeles. El bloque "dilatación" se implementa con el fin de que los taladros restantes se dilaten lo suficiente para restarlos a la imagen original. Las imágenes obtenidas son las siguientes:

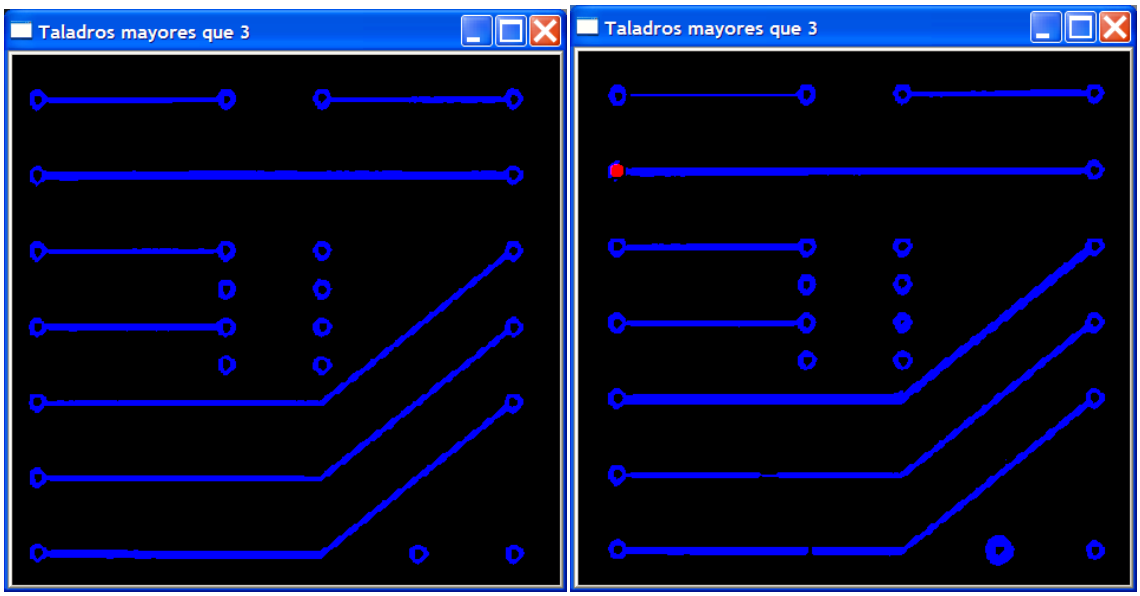



```
element = cvCreateStructuringElementEx(3,3,1,1,CV_SHAPE_CROSS,0);
cvErode(solo_taladros,temp,element,2);
cvDilate(temp,temp2,element,4);
cvReleaseStructuringElement(&element);
cvSub(solo_taladros,temp2,temp,NULL);
cvZero(temp2);
cvMerge(solo_cond,temp2,temp,NULL,color_dst);
```

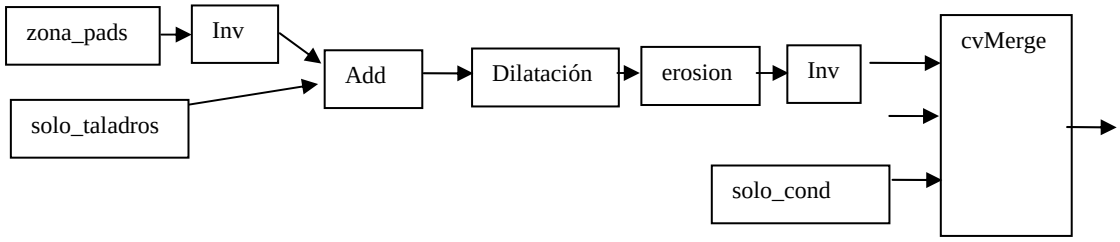




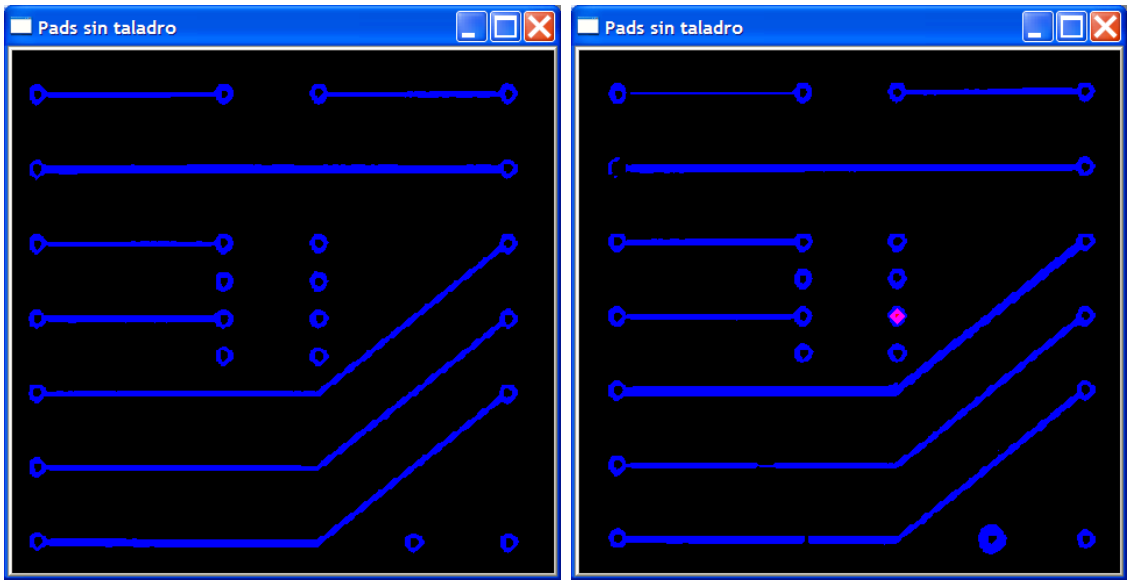
b. Determinación de taladros mayores de 3 pixeles. Se implementa de forma análoga a la anterior *(cambiar parámetros y el bloque "diferencia" por un "and")*. Se pide escribir el diagrama de bloques del algoritmo utilizado e implementar el código para obtener las siguientes imágenes:



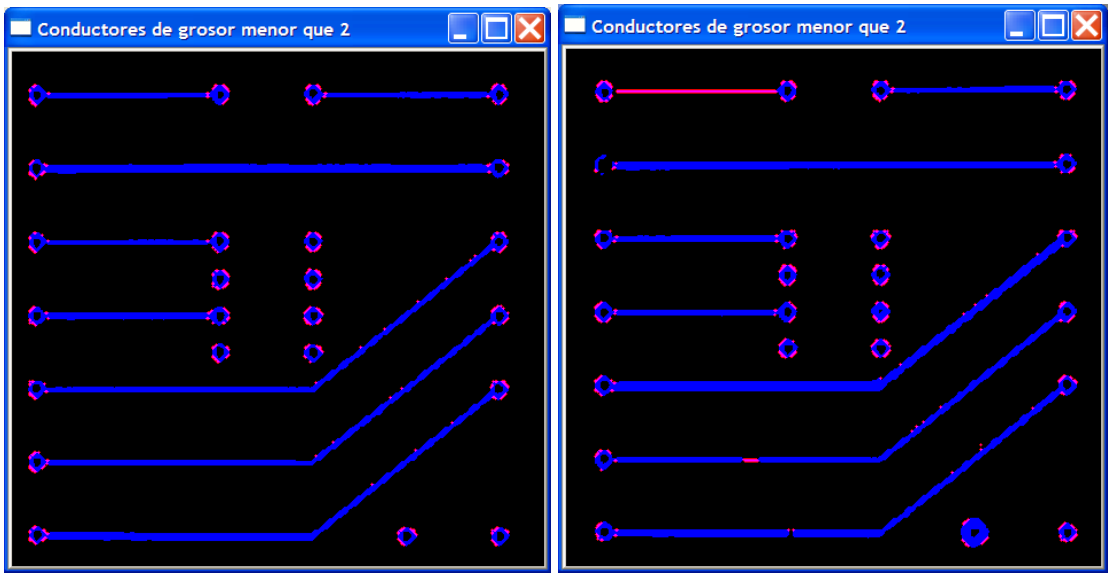
c. Determinación de pads que no poseen taladro. Se muestra el algoritmo utilizado y el código implementado.




```
cvNot(zona_pads,temp);
cvAdd(temp,solo_taladros,temp2,NULL);
element = cvCreateStructuringElementEx(3,3,1,1,CV_SHAPE_CROSS,0);
cvDilate(temp2,temp,element,8);
cvErode(temp,temp2,element,6);
cvReleaseStructuringElement(&element);
cvNot(temp2,temp);
cvZero(temp2);
cvMerge(solo_cond,temp2,temp,NULL,color_dst3);
```



d. Determinación de pistas de grosor menor de 2 pixeles. Una posible implementación es la siguiente: en primer lugar se obtiene una imagen donde solo aparecen pistas mediante la diferencia entre las imágenes **`solo_cond`** y **`zona_pads`**. A la imagen en la que sólo aparecen pistas se le aplica un algoritmo para encontrar las pistas más pequeñas de 2 pixeles, (para ello realizamos una única erosión). Se pide dibujar el diagrama de bloques del algoritmo e implementar el código para obtener las siguientes figuras:



TAF