(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 357 371**

(51) Int. Cl.:
G06T 3/40 (2006.01)
G06T 7/00 (2006.01)
G06K 9/00 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Número de solicitud europea: **08826248 .0**
(96) Fecha de presentación : **07.07.2008**
(97) Número de publicación de la solicitud: **2176829**
(97) Fecha de publicación de la solicitud: **21.04.2010**

(54) Título: **Sistema y método de procesamiento de datos de imagen.**

(30) Prioridad: **05.07.2007 US 948069 P**

(45) Fecha de publicación de la mención BOPI:
25.04.2011

(45) Fecha de la publicación del folleto de la patente:
25.04.2011

(73) Titular/es: **SIEMENS INDUSTRY, Inc.**
3333 Old Milton Parkway
Alpharetta, Georgia 30005-4437, US

(72) Inventor/es: **Horton, David, C.**

(74) Agente: **Carvajal y Urquijo, Isabel**

DESCRIPCIÓN

Sistema y método de procesamiento de datos de imagen

La presente invención hace referencia en general a sistemas de captura y detección de imágenes, y más específicamente a métodos y sistemas para la obtención de datos de imagen.

Antecedentes de la invención

Cada vez más resulta necesaria la utilización sistemas de detección de vehículos en sistemas de control de tráfico. La detección de la presencia y/o cercanía de vehículos permite alcanzar una eficiencia mejorada en el control del tráfico. En un ejemplo simplificado, una señal de tráfico puede no cambiar a verde para un cruce de calles durante un tiempo indefinido si no hay vehículos presentes en el cruce esperando la señal. Sin embargo, si un vehículo se acerca a la intersección en el cruce de calles, la señal de tráfico puede cambiar en respuesta a la detección del vehículo que se acerca.

Históricamente, la detección de vehículos para el control del tráfico ha utilizado sensores inductivos ubicados debajo de la superficie de la vía. La entrada del automóvil en una región cercana al sensor inductivo cambia el campo magnético percibido, permitiendo de este modo la detección del vehículo. Una desventaja de estos sistemas es la necesidad de implantar los sensores inductivos en la parte inferior o en la superficie del pavimento de la vía, lo cual requiere una obra considerable y la interrupción del tráfico. Debido a que las cámaras de vídeo evitan la necesidad de implantar equipos localizados en la parte inferior de la superficie del pavimento de la vía, los sistemas de detección de vehículos basados en cámaras son cada vez más necesitados.

Son conocidos los sistemas de detección de vehículos basados en cámaras que utilizan procesamiento de vídeo. Un tipo de sistemas de detección de vehículos realiza la detección de vehículos en áreas seleccionadas de los fotogramas de vídeo disponibles provistos por una cámara de vídeo. Con este fin, una cámara de vídeo proporciona la información del fotograma de vídeo, habitualmente en un formato conocido como formato Bayer, a un sistema de procesamiento. El sistema de procesamiento convierte el vídeo en formato Bayer a vídeo en formato RGB (del inglés, red-green-blue, rojo-verde-azul), que es (o es similar a) el formato empleado por la mayoría de los sistemas de vídeo digital.

El vídeo con formato RGB se proporciona después al subsistema de detección y opcionalmente a un dispositivo de visualización, tal como por ejemplo un ordenador portátil de ubicación remota. El subsistema de detección emplea después un mapeo por homografía para aislar áreas seleccionadas específicas del fotograma de vídeo para ser utilizadas por los propios algoritmos de detección. En particular, la detección de vehículos habitualmente no ocurre dentro de un fotograma de vídeo entero, sino en áreas aisladas tales como partes de una vía o cerca de objetos específicos. Las áreas deseadas para la detección se convierten entonces en bloques de detección que se estandarizan para los algoritmos de detección.

De este modo, por ejemplo, un área de interés trapezoidal pequeña en o cerca de un cruce se convierte en un rectángulo de un tamaño y orientación apropiados para los algoritmos de detección.

Por lo tanto, la parte de la imagen de vídeo que ha de analizarse se almacena como un *bloque de detección*. Los datos de vídeo correspondientes al área de interés se transfieren al bloque de detección a través de un mapeo por homografía o deformación (warping como se conoce en la técnica). El bloque de detección representa por lo tanto datos de vídeo del área de interés seleccionada.

Los algoritmos de detección operan con los bloques de detección para identificar si los objetos en movimiento están dentro del área de interés seleccionada. Las operaciones de detección incluyen integración, autocorrelación, centrado de las masas cromáticas, procesamiento del fondo, entre otros para detectar los vehículos en movimiento y/o cualquier presencia de vehículos.

La detección algorítmica de vehículo utilizando las técnicas antes mencionadas consume una gran cantidad de potencia de cálculo computacional. Las máquinas de procesamiento que se encuentran habitualmente en el mercado con amplios rangos de temperatura de funcionamiento están bastante limitadas en lo que respecta a su velocidad máxima de procesamiento. Ejecutar el procesamiento de vídeo necesario para la detección de vehículos puede hacer que tales procesadores trabajen “demasiado calientes”.

Por lo tanto, existe una necesidad de procesamiento de vídeo mejorado que satisfaga unas necesidades de detección avanzadas dentro de una potencia de procesamiento limitada.

Resumen de la invención

Se revelan aparatos y métodos que emplean procesamiento de imágenes y que al menos en algunas de las realizaciones abordan las desventajas mencionadas anteriormente del arte previo.

La invención se define mediante las reivindicaciones adjuntas.

Las características y ventajas de la realización descrita anteriormente, además de otras, serán evidentes para los expertos en el arte con referencia a la siguiente descripción detallada y los dibujos adjuntos.

Breve descripción de los dibujos

La figura 1 muestra un diagrama de bloque esquemático de un sistema para el procesamiento de imágenes que incluye al menos una realización de la invención;

La figura 2 muestra un diagrama esquemático representativo de una cámara y un fotograma de imagen;

La figura 3 muestra una representación de vistas bidimensionales del fotograma de imagen de la figura 3 y los correspondientes bloques de detección;

La figura 4 muestra un diagrama de una parte 402 de un fotograma de vídeo en un formato de mosaico;

La figura 5 muestra un diagrama de flujo de un grupo a modo de ejemplo de operaciones que pueden utilizarse para generar un bloque de detección a partir de un bloque de vídeo seleccionado de datos de fotogramas de vídeo en formato de mosaico;

La figura 6 muestra un diagrama de flujo de una operación a modo de ejemplo que puede utilizarse para generar una tabla de consulta según al menos algunas realizaciones de la invención; y

La figura 7 muestra un diagrama de bloque esquemático más detallado de una realización a modo de ejemplo del sistema de procesamiento de imagen de la figura 1.

Descripción detallada

La figura 1 muestra un diagrama de bloque de un sistema 100 para el procesamiento de imágenes que incluye al menos una realización de la invención. El sistema 100 incluye una fuente de datos de fotograma de vídeo 102, un sistema de detección de vehículos 104 y al menos una interfaz remota de usuario 106.

La fuente de datos de fotogramas de vídeo 102 es un dispositivo o conjunto de dispositivos configurado para proporcionar datos de fotogramas de vídeo en un primer formato. El primer formato de datos de fotogramas de vídeo incluye una pluralidad de píxeles, cada píxel contiene datos de un solo color. Por ejemplo, la fuente de datos de fotogramas de vídeo 102 es una cámara digital o una cámara de vídeo digital que emplea un CCD (Dispositivo de carga acoplada, por sus siglas en inglés) de alta velocidad o matriz de generación de imágenes CMOS (Semiconductor óxido-metal complementario, por sus siglas en inglés) sensibles a la luz como sensor inicial. Esta matriz produce valores de datos correspondientes a píxeles individuales (pequeños elementos de imagen individuales) de acuerdo a cuánta intensidad de luz se mida en un píxel particular dentro de la matriz de generación de imágenes. Tales dispositivos son conocidos.

El primer formato de los datos de fotograma de vídeo puede comprender de manera apropiada datos en formato de mosaico. En particular, muchas cámaras de vídeo están configuradas para generar datos de píxeles en mosaico, a veces conocidos como datos con formato Bayer. Las cámaras con formato Bayer habitualmente incluyen lentes de mosaico de colores posicionados frente a una matriz de detectores CCD monocromáticos, y transmite el valor de luminancia para cada elemento en lugar de los componentes de colores individuales. La figura 7, que se discute más adelante, muestra una realización a modo de ejemplo que emplea una cámara con lente de mosaico 130.

Tal fuente transmite de manera efectiva sólo un valor de componente de color por píxel, y para cada cuatro píxeles transferidos, dos son píxeles verdes, uno es un píxel azul y uno es un píxel rojo. Como resultado, la información de vídeo de espectro verde se submuestrea de manera efectiva a una resolución del 50%, mientras que la información roja y azul se submuestrea de manera efectiva a una resolución del 25%. En cualquier caso, los datos de fotograma incluyen una pluralidad de píxeles que tienen solamente información de un solo color.

Con el fin de ofrecer una mayor explicación sobre los datos con formato de mosaico, la figura 4 muestra un diagrama de una parte 402 de un fotograma de vídeo en un formato de un solo color o mosaico. La parte 402 incluye píxeles de un solo color rojos, azules y verdes 404, 406 y 408 respectivamente.

La implementación de la invención no se limita a los formatos de mosaico de un solo color en particular que se muestran en la figura 4. Al menos algunas realizaciones de la invención son compatibles con cada uno de los cuatro formatos Bayer posibles, y pueden adaptarse con facilidad a otros patrones de mosaico de submuestreo similares.

Con referencia nuevamente a la figura 1, la fuente de datos de fotograma de vídeo 102 está acoplada de manera operativa para proporcionar los datos de fotograma de vídeo en el primer formato a cada uno de los sistemas de detección de vídeo 104 y la interfaz de usuario remota 106.

El sistema de detección de vídeo 104 incluye equipos de procesamiento de datos y circuitos asociados que son capaces de realizar procesamiento de imágenes como se describe en la presente invención. El sistema de detección de vídeo 104 está configurado para emplear mapeo por homografía para generar un bloque de detección a partir de los datos de fotograma de vídeo. En esta realización, el bloque de detección es representativo de un subconjunto seleccionado de las imágenes de fotograma de vídeo, y habitualmente incluye una parte de la imagen que ha sido preseleccionada como imagen que potencialmente contiene vehículos a ser detectados. El mapeo por homografía (o warping) se emplea para generar una vista re-alineada, por ejemplo, un rectángulo generado a partir de un

trapezoide, de modo tal que los puntos de vista angulares se rectifican para convertise en una vista normal (es decir, perpendicular) que ayude a los algoritmos de detección.

En una aplicación alternativa, un sistema de detección de vehículos puede utilizarse en un sistema de seguridad u otro sistema de vigilancia que emplea cámaras giratorias. En tal caso, el mapeo por homografía permite comparar varios puntos de vista de una misma cámara en movimiento (por ejemplo, rotando) y procesarlos a partir del mismo punto de vista. En esta realización alternativa, el sistema de detección de vehículos puede estar conectado para recibir los datos de fotograma de vídeo de las cámaras utilizadas para la vigilancia general. Las cámaras de vigilancia general a menudo están configuradas para girar y así cubrir un área más amplia.

Con referencia nuevamente a la figura 1, la fuente de datos de vídeo 102 en esta realización habitualmente es un sistema de captura de imagen fijo. En este caso, el mapeo por homografía se utiliza para mapear partes seleccionadas (y formas no estándares) de los datos de vídeo en un bloque de detección. El bloque de detección tiene un tamaño y una forma (cuadrado o rectangular) compatible con los algoritmos de detección. La parte seleccionada de los datos de vídeo a menudo es seleccionada por el usuario y usualmente tendrá al menos algunas dimensiones (tamaño, forma) que difieren de las del bloque de detección, y habitualmente será significativamente mayor.

Por ejemplo, las figuras 2 y 3 ilustran a modo de ejemplo cómo se emplea el mapeo por homografía para generar representaciones consistentes y normalizadas de partes seleccionadas (y con diferentes formas y tamaños) de una imagen de fotograma de vídeo. La figura 2 muestra una cámara 202 (que puede utilizarse de manera apropiada como la fuente de información de vídeo 102 de la figura 1), y un fotograma de imagen 204. El fotograma de imagen 204 consiste en el campo de visión bidimensional de la cámara 202. En este ejemplo, un usuario ha definido dos áreas de interés 206, 208 en el fotograma de imagen. Por ejemplo, estas áreas de interés 206, 208 pueden incluir enfoques a distintas partes de un cruce de tráfico. Como puede verse en la figura 2, las áreas de interés 206, 208 se encuentran en diferentes ubicaciones dentro del fotograma de imagen 204, tienen diferentes formas y diferentes tamaños.

La figura 3 muestra una representación de vistas bidimensionales del fotograma de imagen 204, las áreas de interés o partes de la imagen 206, 208 y los correspondientes bloques de detección 210, 212. Las técnicas de mapeo por homografía se utilizan para mapear las partes de imagen 206, 208 a bloques de detección 210, 212 de tamaño y orientación similares. Los algoritmos de detección, no se muestran, operan con los bloques de detección 210, 212. Por lo tanto, pueden utilizarse los mismos algoritmos de detección para una variedad de partes de imagen que tienen un tamaño, forma y orientación únicos. Tales técnicas de mapeo por homografía son conocidas.

Con referencia nuevamente a la figura 1, el sistema de detección de vehículos 104 está configurado para realizar el mapeo por homografía de una imagen de interés de cada fotograma de vídeo recibido de la cámara de vídeo 202, aunque los datos de fotograma de vídeo siguen estando en formato de un solo color. El sistema de detección de vehículos 104 está configurad para convertir los píxeles de formato de un solo color del vídeo de entrada en píxeles de múltiples colores (por ejemplo, rojo, verde, azul) del bloque de detección.

Con este fin, el sistema de detección de vehículos 104 asocia cada píxel del bloque de detección con los datos de píxeles de múltiples píxeles de los datos de fotograma de vídeo. Cada píxel del bloque de detección se asocia con datos de píxeles de múltiples píxeles con datos de colores diferentes, de modo tal que cada píxel del bloque de detección tiene información representativa de múltiples colores. Con referencia a la figura 4, por ejemplo, un solo píxel del bloque de detección puede contener información de cada uno de un píxel rojo 404, un píxel azul 406 y un píxel verde 408.

En cualquier caso, el bloque de detección resultante sólo contiene información de un subconjunto pequeño del fotograma de vídeo original, y contiene píxeles que se han convertido o interpolado al formato de múltiples colores.

Por lo tanto, el sistema de detección de vídeo 104 está configurado para convertir los datos de vídeo en formato de un solo color de una parte (y sólo una parte) de los datos de fotograma de vídeo original que corresponden al bloque de detección en datos de vídeo formateados de múltiples colores.

A diferencia del arte previo, el sistema de detección de vídeo 104 sólo realiza el demosaico de la parte de los datos de fotograma de vídeo que corresponde al bloque de detección seleccionado en lugar de todo el fotograma de vídeo. Esto ahorra potencia computacional ya que los esquemas de interpolación utilizados para el demosaico consumen muchos recursos computacionales.

Con referencia en general nuevamente a la figura 1, el sistema de detección de vídeo 104 también está configurado para realizar algoritmos de detección de vehículos de uno o más bloques de detección generados como se describe más arriba. La detección de vehículos en un bloque de vídeo puede lograrse utilizando varias técnicas conocidas en el arte, tales como integración, autocorrelación, centrado de las masas cromáticas, procesamiento del fondo y similares.

Como se discutió más arriba, la interfaz de usuario remota 106 también está configurada para recibir los datos de fotograma de vídeo en el primer formato. La interfaz de usuario remota 106 puede ser un dispositivo informático de uso general o de uso específico. La interfaz de usuario remota 106 está configurada para realizar el demosaico

(conversión entre datos de píxel de un solo color de diferentes colores a datos de píxeles de múltiples colores) en todo el fotograma de imagen de los datos de fotograma de vídeo y mostrar los datos de fotograma en una pantalla o visualizador de vídeo. A continuación se proporcionan más detalles sobre el demosaico por parte de la interfaz de usuario 106 con referencia a la figura 7.

Por lo tanto, el sistema de procesamiento de imágenes 100 antes descrito proporciona datos que pueden utilizarse tanto para la visualización (a través de la interfaz de usuario 106) como para la detección de vehículos (a través del sistema de detección 104) mientras se reduce la carga computacional en el circuito de procesamiento de detección de vehículos.

La figura 7 muestra un diagrama de bloque esquemático más detallado de una realización a modo de ejemplo del sistema de procesamiento de imágenes 100 de la figura 1.

En la realización de la figura 7, la fuente de datos de fotograma de vídeo 102 es una cámara digital 130 o grabador de vídeo digital que emplea una matriz 132 de detectores CCD monocromáticos de alta velocidad. Esta matriz 132 produce valores de datos correspondientes a píxeles individuales (pequeños elementos de imagen individuales) según cuánta intensidad de luz se mida en ese píxel particular dentro de la matriz de generación de imágenes 132. La cámara 130 habitualmente también incluye una lente de mosaico de colores 134 posicionado frente al matriz 132. La lente de mosaico de colores 134, como se conoce en el arte, filtra la luz de modo tal que el valor de luminancia para cada píxel representa la intensidad de un componente de color individual específico. La salida de la cámara 130 puede ser datos con formato Bayer tal como se muestra en la figura 4, y se describe en general en la patente estadounidense Nº 3971065, que se incorpora a la presente a modo de referencia. Como con la figura 1, los datos producidos por la cámara 130 son datos de fotograma de vídeo en el primer formato. Cada conjunto de datos de fotograma de vídeo consiste en un solo fotograma de vídeo, que es una imagen fija.

La cámara 130 se acopla en modo operativo para proporcionar los datos de fotograma de vídeo en el primer formato de cada sistema de detección de vídeo 104 y la interfaz de usuario remota 106.

El sistema de detección de vídeo 104 incluye un circuito de procesamiento 138 que incluye al menos una memoria 136 y un circuito asociado que es capaz de realizar los pasos 140, 142, 144 y 146 como se describe en la presente invención. El circuito de procesamiento 138 está configurado para recibir los datos de fotograma de vídeo en el primer formato y generar (paso 140) datos de píxeles de un bloque de detección representativo de un subconjunto de los datos de fotograma de vídeo en los cuales ha de realizarse la detección de imágenes. Para generar el bloque de detección, el circuito de procesamiento 138 utiliza una tabla de consulta 148 (almacenada en la memoria 136) para asociar los datos de píxel de múltiples píxeles de los datos de fotograma de vídeo de un solo color con cada píxel del bloque de detección.

El bloque de detección generado comprende datos de píxeles representativos de una parte seleccionada de los datos de imagen de vídeo de entrada. Los datos de píxel para cada píxel del bloque de detección generado tienen múltiples componentes de colores. La figura 5, que se discute más abajo, proporciona un conjunto de pasos a modo de ejemplo que realizan la función del paso 140 como se describe más arriba.

El circuito de procesamiento 138 también se opera para realizar la detección de imágenes. En particular, en el paso 142, el circuito de procesamiento realiza algoritmos de detección de vehículos en uno o más bloques de detección generados como se describe más arriba. La detección de vehículos en un bloque de vídeo puede lograrse utilizando varias técnicas conocidas en el arte, tales como integración, autocorrelación, centrado de las masas cromáticas, procesamiento del fondo y similares. Tales técnicas de detección varían y son conocidas por los expertos en el arte.

A partir de allí, en el paso 144, el circuito de procesamiento 138 realiza un control en base a la detección de imagen. A modo de ejemplo, puede alterarse una operación de señal de tráfico como resultado de una detección de la presencia de un vehículo en uno o más bloques de detección generados en el paso 140. En otro ejemplo, la ausencia de movimiento en uno o más áreas monitoreadas puede resultar en la alteración de la operación de señal de tráfico. Varios métodos de control de tráfico basados en la presencia o movimiento de vehículos detectados son conocidos.

Para realizar los pasos anteriores 140, 142 y 144, y en particular el paso 140, la tabla de consulta 148 debe generarse para mapear partes de imagen de usuario seleccionadas al bloque detector. En esta realización, el circuito de procesamiento 138 se opera para generar la tabla de consulta 148 utilizada para esta generación de bloque de detección. En particular, el circuito de procesamiento 138 puede recibir entradas de manera apropiada identificando partes seleccionadas de una imagen (por ejemplo, partes de imagen 206, 208 de la figura 2) en la cual la detección de imagen ha de tener lugar a través del paso 142. La parte de imagen puede tener forma de cuadrado, rectángulo, trapezoide, paralelogramo u otra forma, y puede ser de diversos tamaños y tener distintas ubicaciones dentro del fotograma de vídeo. La entrada puede recibirse de la interfaz de usuario 106.

El circuito de procesamiento 138 utiliza después mapeo por homografía, entre otras cosas, para crear una tabla de consulta que asocia cada píxel de un bloque de detección con múltiples píxeles de un solo color de un fotograma de vídeo de entrada que corresponde a la parte de imagen seleccionada por el usuario. Los múltiples píxeles de un solo color preferentemente incluyen diferentes píxeles de colores del fotograma de vídeo de entrada de modo tal que

cada píxel del bloque de detección puede tener información de múltiples colores. La figura 6, que se discute más abajo, muestra una operación a modo de ejemplo en donde puede generarse una tabla de consulta 148 tal.

Con referencia nuevamente a la figura 7, como se discutió más arriba, la interfaz de usuario remota 106 también está configurada para recibir los datos de fotograma de vídeo en el primer formato. La interfaz de usuario remota 106 en esta realización es un ordenador portátil (por ejemplo, una laptop) que se opera conectado a la cámara 130 y al circuito de procesamiento 138. El ordenador 106 está configurado y/o programado para realizar los pasos aquí descritos. El ordenador 106 se utiliza mayormente para la configuración y mantenimiento de la cámara 130 y/o el circuito de procesamiento 138, y por lo tanto no es necesario que esté unido de manera permanente al sistema 100.

Como se discutió más arriba en relación con la figura 1, la interfaz de usuario remota/el ordenador 106 está configurada para realizar el demosaico (conversión entre datos de píxeles de un solo color de diferentes colores a datos de píxeles de múltiples colores) en todo el fotograma de imagen de los datos de fotograma de vídeo y mostrar los datos de fotograma en una pantalla de vídeo o visualizador. La velocidad de imagen de visualización en el ordenador 106 puede ser de manera apropiada de sólo 100 imágenes por minuto, lo cual permite el proceso de demosaico en dispositivos informáticos personales normales.

Se conocen varias operaciones de demosaico, incluyendo interpolación por aproximación, como una interpolación lineal, como una interpolación bilineal, como una interpolación por splines cúbicos, o como un tipo diferente de interpolación spline, normalmente con mejor calidad visual otorgada por interpolaciones más complejas. En algunas realizaciones, el ordenador 106 puede configurarse para seleccionar de una pluralidad de operaciones de demosaico, y velocidades de imagen de modo tal que el balance entre velocidad de imagen, precisión y complejidad y potencia de procesamiento informático disponible en la interfaz de usuario/ordenador 106 puede ajustarse de manera selectiva.

Como se discutió más arriba, durante la operación normal de un sistema configurado, las tablas de consulta 148 para una o más partes de imagen seleccionadas por el usuario ya han sido generadas y almacenadas en la memoria 136.La operación normal consiste en los pasos 140, 142 y 144 como se describiera con anterioridad.

Con respecto al paso 140, la figura 5 muestra en más detalle las operaciones que se utilizan para generar un bloque de detección de un bloque de vídeo seleccionado de datos de fotograma de vídeo que está en un formato de mosaico.

En el paso 502, los datos de fotograma de vídeo en el primer formato se reciben de la fuente de datos de fotograma de vídeo 102. Como se discutiera con anterioridad, los datos de fotograma de vídeo están en un formato de un solo color (mosaico).

Los pasos 504 a 510 representan un bucle for que se indexa a través de cada píxel del bloque de detección. Por lo tanto, en el paso 504, el siguiente píxel del bloque de detección se indexa. Esto se indica de aquí en adelante como píxel actual.

En el paso 506, la ubicación del píxel mismo dentro del bloque de detección se indexa a la tabla de consulta 148. Como se discutiera con anterioridad, la tabla de consulta 148 asocia cada píxel del bloque de detección con una pluralidad de píxeles dentro de los datos de fotograma de vídeo. Esta pluralidad de píxeles preferentemente incluye al menos un píxel verde, al menos un píxel azul y al menos un píxel rojo de los datos de fotograma de vídeo en mosaico. En el paso 508, los datos de píxel de los píxeles de los datos de fotograma de vídeo asociados con el píxel actual se obtienen de los datos de fotograma de vídeo y se almacenan como los datos de píxel para el píxel actual del bloque de detección.

En el paso 510, se determina si hay píxeles adicionales en el bloque de imagen. De ser así, el siguiente píxel se identifica en el paso 504 y el proceso se repite en consecuencia. De no ser así, se ha creado todo el bloque de detección, y se realiza el paso 512. En el paso 512, el bloque de detección se almacena en la memoria, no se muestra, para su utilización por parte de las operaciones de detección de imagen 142.

Las operaciones antes descritas representan una mejora sobre el arte previo en tanto que el sistema de detección no necesita interpolar un fotograma de vídeo entero. Sólo una parte seleccionada de los datos de fotograma de vídeo se interpola de manera efectiva. En esta realización, se utiliza una tabla de consulta 148 para generar cada píxel del bloque de detección a partir de píxeles de diferentes colores de los datos de fotograma de vídeo en mosaico. La tabla de consulta 148 también realiza el mapeo por homografía dado que la selección de la ubicación de píxeles de los datos de fotograma de vídeo que corresponden a los píxeles del bloque de detección se identifica a través del mapeo por homografía. Por lo tanto, se requiere el desarrollo de una tabla de consulta apropiada para realizar las operaciones descritas con anterioridad.

Por lo tanto, la figura 6 muestra un grupo de operaciones a modo de ejemplos que pueden utilizarse para desarrollar una tabla de consulta que se utiliza para realizar el paso 506 de la figura 5 y el paso 140 de la figura 7. En particular, la figura 6 muestra en detalle un conjunto de operaciones que realizan el paso 146 de la figura 7. Además, la tabla de software proporcionada a continuación incluye códigos para desarrollar la tabla de consulta. En esta realización, la tabla de consulta 148 se genera mediante el circuito de procesamiento 138 del sistema de detección 104. En otras realizaciones, el equipo de procesamiento en la interfaz de usuario 106 puede realizar los pasos de la figura 6 para

generar la tabla de consulta 148, que se transferiría y almacenaría después en la memoria 148. Sin embargo, en la presente realización, las operaciones de la figura 6 se realizan mediante el circuito de procesamiento 138 del sistema de detección de imagen 104.

En el paso 602, el circuito de procesamiento 138 recibe datos de entrada que define un bloque de imagen de fotograma de vídeo a ser mapeada en un bloque de detección. Tal bloque de imagen de fotograma de vídeo constituye una parte del campo de imagen de la cámara en la cual ha de detectarse y/o monitorearse la ubicación o presencia de un vehículo. Por ejemplo, el circuito de procesamiento 138 puede recibir de la interfaz de usuario 106 información que identifica partes de imagen 206 y/o 208 (véase las figuras 2 y 3) para monitoreo. El circuito de procesamiento 138 recibe la información que identifica partes de imagen seleccionadas de cualquier manera apropiada, tal como por identificación de los píxeles de esquina de la parte de imagen.

Cabe destacar que el usuario puede definir múltiples partes de imagen, tales como las partes 206 y 208 de las figuras 2 y 3, para monitoreo. Los pasos 604 a 616 de la figura 6 se realizan de manera separada para cada parte de imagen identificada. Por lo tanto, por ejemplo, si las partes de imagen 206 y 208 de las figuras 2 y 3 son seleccionadas por el usuario, los pasos 604-616 se realizan para preparar una tabla de consulta para la parte 206, y los pasos 604-616 también se realizan para preparar una tabla de consulta separada para la parte 208.

Como paso inicial 604, las ecuaciones de mapeo por homografía se resuelven para la parte de imagen seleccionada y el bloque de detección. Las ecuaciones de mapeo por homografía asocian el bloque de detección (por ejemplo, el bloque 210), con una parte de un fotograma de imagen (por ejemplo, la parte 206 de la figura 3). El bloque de detección y la parte de imagen seleccionada normalmente tienen un tamaño y/o forma diferentes. La parte de imagen seleccionada también está inserta dentro de un fotograma de vídeo más grande (por ejemplo, el fotograma 204 de la figura 3), mientras que el bloque de detección no lo está. Resolver tales ecuaciones de mapeo por homografía implica en este caso ocho incógnitas y ocho ecuaciones. Tales ecuaciones homográficas y sus soluciones se conocen, y se proporciona un ejemplo en el código de software adjunto.

Las ecuaciones homográficas resultantes ayudan a proporcionar una traslación de una ubicación en el bloque de detección a una ubicación en los datos de imagen de vídeo.

En los pasos 606 a 614, la tabla de consulta 148 se completa píxel por píxel, utilizando la solución de homografía del paso 604.

En el paso 606, se indexa el siguiente píxel en el bloque de detección. Este píxel se indica a continuación como el píxel actual. En el paso 608, el circuito de procesamiento 138 utiliza la solución de homografía del paso 604 para asociar el píxel actual con un píxel del fotograma de vídeo dentro de la parte de imagen seleccionada. Por lo tanto, por ejemplo, si el píxel actual es la esquina superior izquierda del bloque de detección, la solución de homografía 604 probablemente asociaría el píxel actual con la esquina superior izquierda de la parte de imagen seleccionada.

Como resultado, la solución de homografía se utiliza en el paso 608 para identificar un píxel del fotograma de vídeo (dentro de la parte de imagen seleccionada) que corresponde con el píxel actual. El píxel identificado se indica como *píxel correspondiente*. El píxel correspondiente en cada conjunto de datos de fotograma de vídeo contendrá datos de píxeles que tienen información de un solo color.

En el paso 610, el circuito de procesamiento 138 identifica y selecciona píxeles vecinos que están configurados para tener otros colores, uno solo cada uno. Por ejemplo, si el píxel correspondiente es un píxel rojo, el procesador en el paso 610 identifica píxeles vecinos azules y verdes. Por lo tanto, el píxel correspondiente y los píxeles vecinos deben incluir información para todos los píxeles de colores del formato de datos de imagen de vídeo de salida.

Preferentemente, los dos píxeles vecinos se seleccionan de modo tal que incluyen al menos un píxel de otra línea horizontal de modo tal que el píxel correspondiente y los dos píxeles vecinos forman un grupo. Por ejemplo, con referencia a figura 4, si el píxel correspondiente se determina en el paso 608 como un píxel verde 408, el procesador puede de manera apropiada seleccionar el píxel rojo 404 y el píxel verde 406 como los píxeles vecinos. Cabe destacar que en datos en mosaico con formato Bayer (véase la figura 4), se necesitan datos de píxeles de al menos dos líneas para obtener una representación de todos los colores de píxeles disponibles.

En el paso 612, se crea una entrada en la tabla para el píxel actual. La entrada en la tabla identifica las ubicaciones del píxel correspondiente determinado en el paso 608 y los píxeles vecinos determinados en el paso 610. Después del paso 612, la tabla de consulta 148 se ha completado para al menos el píxel actual. En esta realización, la tabla de consulta 148 identifica tres píxeles de un solo color de los datos de fotograma de vídeo de entrada que se utilizan para formar el píxel actual del bloque de detección.

En el paso 614, el procesador determina si una entrada en la tabla debe determinarse para más píxeles en el bloque de detección. De ser así, el procesador vuelve al paso 606 para procesar el siguiente píxel del bloque de detección. De no ser así, sin embargo, el procesador pasa al paso 616.

En el paso 616, el procesador ha completado la operación de creación de la tabla de consulta. La tabla de consulta 148 proporciona ahora una traslación de un bloque de imagen en mosaico que tienen un primer conjunto de

dimensiones a un bloque de imagen interpolado (el bloque de detección) que tiene un segundo conjunto de dimensiones, incluyendo una forma preferente de cuadrado o rectángulo.

La Tabla I muestra un código de software a modo de ejemplo utilizado para generar la tabla de consulta 148. La Tabla II muestra un código de software para generar un bloque de detección utilizando la tabla de consulta 148.

Tabla I

```
typedef struct pixel_idx6 {          // tabla de consulta de seis dimensiones
                        point red;
                        point green;
                        point blue;
} pixel_idx6;
typedef struct pixel_idx3 {          // Nueva versión de la tabla de consulta
                        unsigned int redidx;
                        unsigned int greenidx;
                        unsigned int blueidx;
 } pixel_idx3;
typedef enum {REDPIX=0, GREEN1PIX=1, GREEN2PIX=2, BLUEPIX=3} BAYER_RGB;
//-------------------------------------------------------------------------------------------------------
// int Set(tl, bl, br, tr, w=0, h=0) – Configurar – Configurar una nueva línea.
// Entradas:
// Cpoint tl – Punto superior izquierdo con respecto al rectángulo rectificado
// Cpoint bl – Punto inferior izquierdo con respecto al rectángulo rectificado
// Cpoint br – Punto inferior derecho con respecto al rectángulo rectificado
// Cpoint tr – Punto superior derecho con respecto al rectángulo rectificado
//     w     - ancho definido por el usuario. Se fija en 0 para definición automática
//     h     - altura definida por el usuario. Se fija en 0 para definición automática
//
// Procesamiento:
// 1.                   El ancho por defecto es el promedio del ancho inferior/ inferior.
//                      La altura por defecto es el promedio de la altura izquierda/derecha.
// 2.                   La homografía se computa y almacena.
//
// Valor de retorno:
//                      0 – OK
//      -1 – No se pudo computar homografía (matriz individual).
//-------------------------------------------------------------------------------------------------------
int Clane:: Set(CPoint tl, CPoint bl, CPoint br, CPoint tr, unsigned w, unsigned h)
{
 CDoc *pDoc= GetDocument();
 m_TopLeft = tl;
 m_TopRight = tr;
```

```
m_BotLeft= bl;
m_BotRight = br;
m_Width = (w==0 ? (unsigned) (((br.x-bl.x)+(tr.x-tl.x))/2): w);
m_Height = (h==0 ? (unsigned) (((bl.y-tl.y)+(br.y-tr.y))/2): h);
if (ComputeHomography() == -1) return -1;
// Se encontró homografía, computar tabla de consulta
if (m_LookUpTable ! = NULL) {
                    delete [] m_LookUpTable;
                    m_ LookUpTable = NULL;      //
                    printf (“RGB index table deleted\n”);
}
if (dmLookUpTable ! = NULL) {
                    delete [] dmLookUpTable;
                    dm LookUpTable = NULL;      //
                    printf (“RGB index table deleted\n”);
}
int colorid = pDoc ->m_colorid;
// crear ambas tablas de interpolación para el presente
try {
  m_LookUpTable = new CPointS [RECT_WIDTH * RECT_HEIGHT]; //
}
catch (…) {
   m_ LookUpTable= NULL;
   return -1;
}
printf (“Creating RGB index table\n”);
try {
  dmLookUpTable = new pixel_idx3[RECT_WIDTH * RECT_HEIGHT];
}
catch (…) {
   dm LookUpTable= NULL;
   return -1;
}
printf (“Creating Bayer index table\n”);
unsigned pixNum= 0;
for (unsigned y=0; y<RECT_HEIGHT; y++){
   for (unsigned x=0; x<RECT_WIDTH; x++, pixNum++) {
                    float denom = x*m_Hom[6] + y*m_Hom[7] + 1;
                    float nx = (x*m_Hom[0] + y*m_Hom[1] +m_Hom[2]) / denom + 0.5f;
```

```
                    float ny = (x*m_Hom[3] + y*m_Hom[4] +m_Hom[5]) / denom + 0.5f;

                    short ix = (short) nx;
                    short iy = (short) ny;
                    if (ix < 0) ix = 0;
                    if (iy < 0) iy = 0;
                    if (ix >= (short)pDoc->m_camImgWidth) ix = (short)pDoc->m_camImgWidth-1;
                    if (iy >= (short)pDoc->m_camImgHeight) iy = (short)pDoc->m_camImgHeight-1;
# if 0
                    if (colorid ! = LUCAM_COLOR_FORMAT_BAYER_RAW) {
                      m_LookUpTable[y*RECT_WIDTH+x] = CPointS(ix,iy);
} else {
                    WarpIndices(dmLookupTable[y*RECT_WIDTH+x],ix,iy);
}
#else
      m_LookUpTable[y*RECT_WIDTH+x] = CPointS(ix,iy);
      WarpIndices(dmLookupTable[y*RECT_WIDTH+x],ix,iy);
#endif
}
}
  printf("%sDemosaicing table built correctly!\n",(colorid !=
LUCAM_COLOR_FORMAT_BAYER_RAW)? "RGB" : "Bayer");
  return (int)pixNum;
}

//-------------------------------------------------------------------------------------------
// int ComputarHomografía() – Computar Homografía plana.
// Ingresar parámetros:
//                    ninguno – utilizar sólo miembros de la clase.
//
// Procesamiento:
// Las funciones configuran una ecuación linear de 8x8, resolver para homografía
// y se almacena. La homografía se computa a partir de un rectángulo rectificado
// *hacia atrás* al trapezoide para permitir warping (deformación) sin problemas utilizando interpolación.
//
// Valor de retorno:
//                    0 – OK
//    -1 – No se pudo computar homografía (matriz individual).
//-------------------------------------------------------------------------------------------
```

```
int Clane::ComputeHomograhpy()
{
 int i;
float X [4] = (0,0,m_Width, m_Width); // rectángulo rectificado coordinada x en el sentido contrario a las agujas del reloj
float Y [4] = (0,0,m_Height, m_ Height, o); // rectángulo rectificado coordinada y
float Xtag [4] = {m_TopLeft.x,m_BotLeft.x,m_BotRight.x,m_TopRight.x}; // región de imagen coordinadas x
float Ytag [4] = {m_TopLeft.y,m_BotLeft.y,m_BotRight.y,m_TopRight.y}; // región de imagen coordinadas y
// configurar un sistema de ecuaciones
float M[8][8];
float B[8];
float * H=m_Hom;
memset(M,0,sizeof(M));
memset(B,0,sizeof(B));
memset(H,0,sizeof(H));
for (i=0; i<4; i++){
   M[2*i][0] =X[i];
   M[2*i][1] =Y[i];
   M[2*i][2] =1;
   M[2*i][3] =0;
   M[2*i][4] =0;
   M[2*i][5] =0;
   M[2*i][6] = - X[i]* Xtag[i];
   M[2*i][7] = - Y[i]* Xtag[i];
   M[2*i+1][0] =0;
   M[2*i+1][1] =0;
   M[2*i+1][2] =0;
   M[2*i+1][3] = X[i];
   M[2*i+1][4] = Y[i];
   M[2*i+1][5] = 1;
   M[2*i+1][6] = - X[i]* Xtag[i];
   M[2*i+1][7] = - Y[i]* Xtag[i];
   B[2*i] = Xtag[i];
   B[2*i+1][7] = Ytag[i];
}
  return solve_equations(&M[0][0], B, H, 8);
}
//=============================================================================================
// resolver_ecuaciones – resolver system
```

```
//=============================================================================================
int solve_equations( float *mat, float *vect, float *var, int n )
 {
                int i, j, k;
                float fact, temp;
                assert(n >=2) {
                    return solve_2_equations(mat, vect, var);
                }
                if (n==3) {
                    return solve_3_equations(mat, vect, var);
                }
                for ( I = 0; 1< (n-1) ; i ++) {
                    if ( mat[i *n+1] ==0) {
                            for (j = i ; ((mat[j*n+i] == 0) && (j < n) ); j++ ) ;
                            if (j==n)  {
                                    return – 1;
                            }
                            for ( k= 0; k< n; k ++ ) {
                                    temp = mat[i* n+k];
                                    mat[i* n+k] = mat[j* n+k];
                                    mat[j* n+k] = temp;
                            }
                            temp = vect[i];
                            vect[i] = vect[j];
                            vect[j] = temp;
                    }
                    for ( j = I + 1; j < n; j ++ ) {
                            fact = mat[j* n+i]/ mat[i* n+i];
                            for ( k= 0; k< n; k ++ )
                                    mat[j* n+k] -= (fact * mat[i* n+k];
                            vect[j] -= (fact * vect[i])<
                }
        }
                for ( i = (n -1); i > 0; i -- )
                    for ( j = 0; j < i; j ++ ) {
                    if ( mat[i*n+i] == 0.0)
                            fact = mat[j*n+i] / EPSILON_S;
                    else
                            fact = mat[j*n+i] / mat[i*n+i];
```

```
                    for ( k = 0; k < n; k ++ )
                        mat[j*n+k] -=(fact * mat[i*n+k]);
                    vect[j] -= (fact * vect[i]);
                    }
                    for ( i = 0; i < n; i ++ )
                        if ( mat[i*n+i] == 0.0)
                            var[i] = vect[i] / EPSILON_S;
                    else
                        var[i] = vect[i] / mat[i*n+i];
                    return 0;
}
//-------------------------------------------------------------------------------------------------
// WarpIndices(&curPix, ix, iy) nulo – Warping de los índices ya calculados
//                    de la tabla de mapeo por homografía al índice en
//                    la imagen de fotograma raw en Bayer directamente y extraer
//                    los coeficientes de colores para cada píxel en el
//                    bloque de detección
// Parámetros de entrada:
//
// fRgbImg &dst – imagen de destino
//     &curPix – pointer  a la entrada de 3 elementos en la tabla de mapeo
//                    que corresponde a este píxel del bloque de detección
//     ix – índice x en el fotograma de entrada, antes de la deformación (warping)
//     iy – índice y en el fotograma de entrada, antes de la deformación (warping)
//
// Ningún valor devuelto mediante retorno
// Se calculan seis coeficientes y se almacenan en la tabla de mapeo
//
//-------------------------------------------------------------------------------------------
void Clane::WarpIndices(pixel_idx3 &curPix, int ix, int iy)  {
                    unsigned int width, topleftpix;
                    bool firstLine, firstCol;
                    bool blueLine, greenPix;
                    CDoc * pDoc = GetDocument();
                    width = pDoc ->m_camImgWidth;
                    topleftpix= pDoc ->m_topleftpix;
                    firstLine = ((ix < width) && (iy==0)) ? 1 : 0;
    firstCol = (ix % width) ? 0 : 1;
    blue Line = greenPix = 0;
```

```
    if ((topleftpix == GREEN2PIX)) || (topleftpix == BLUEPIX))
                        blueLine = 1;
    if ((topleftpix == GREEN1PIX)) || (topleftpix == GREEN2PIX))
                        greenPix = 1;
    if (iy & 0x01) {
                        blueLine = (blueLine ? 0 : 1);
                        greenPix = (greenPix ? 0 : 1);
  // ? blueLine = !blueLine;
  // ? greenPix = !greenPix;
    }
    if (ix & 0x01) {
                        greenPix = (greenPix ? 0 : 1);
    }
    if (firstLine && firstCol)  {
                        if (blueLine) {
                            if (greenPix) {
                                    curPix.redidx = width;
                            curPix.greenidx = 0;
                            curPix.blueidx = 1;
                            }
                    else {
                            curPix.redidx = width +1;
                            curPix.greenidx = 1;
                            curPix.blueidx = 0;
                            }
            }
            else {
                    if (greenPix) {
                            curPix.redidx =1;
                            curPix.greenidx = 0;
                            curPix.blueidx = width;
                            }
                    else {
                            curPix.redidx = 0;
                            curPix.greenidx = 1;
                            curPix.blueidx = width +1;
                    }
            }
  }
```

```
else if (firstLine) {
                    if (blueLine) {
                if (greenPix) {
                        curPix.rediдx = width + ix;
                        curPix.greenidx = ix;
                        curPix.blueidx = ix - 1;
                        }
                else {
                        curPix.rediдx = width + ix - 1;
                        curPix.greenidx = ix - 1;
                        curPix.blueidx = ix;
                    }
                }
                else }
                  if (greenPix) {
                        curPix.rediдx = ix - 1;
                        curPix.greenidx = ix;
                        curPix.blueidx = width + ix;
                }
                else {
                        curPix.rediдx = ix;
                        curPix.greenidx = ix - 1;
                        curPix.blueidx = width + ix -1;
                        }
                    }
}
else if (firstCol) {
        if (blueLine) {
                if (greenPix) {
                        curPix.rediдx = iy*width - width;
                        curPix.greenidx = iy * width;
                        curPix.blueidx = iy* width + 1;
                        }
                else {
                        curPix.rediдx = iy*width +1 - width;
                        curPix.greenidx = iy*width + 1;
                        curPix.blueidx = iy*width;
                    }
                }
```

```
                    else {
                      if (greenPix) {
                            curPix.redidx = iy*width + 1;
                            curPix.greenidx = iy*width;
                            curPix.blueidx = iy*width - width;

                                  }
                            }
                        }
                        else {          // por mucho la situación más probable
                        if (blueLine) {
                      if (greenPix) {
                            curPix.redidx = iy*width + ix - width;
                            curPix.greenidx = iy * width + ix;
                            curPix.blueidx = iy* width + ix - 1;
                            }
                    else {
                            curPix.redidx = iy*width +ix – width - 1;
                            curPix.greenidx = iy*width + ix - 1;
                            curPix.blueidx = iy*width + ix;
                    }
            }
            else {
                      if (greenPix) {
                            curPix.redidx = iy*width + ix - 1;
                            curPix.greenidx = iy*width + ix;
                            curPix.blueidx = iy*width + ix - width;
                    }
                    else {
                            curPix.redidx = iy*width + ix;
                            curPix.greenidx = iy*width + ix - 1;
                            curPix.blueidx = iy*width + ix – 1 - width;
                      }
                    }
            }
    }
```

Tabla II

```
//-------------------------------------------------------------------------------------------------
// RectifyBayer(src, dst) nulo – Rectificar trapezoide en rectángulo
```

```
//                                  utilizando deformación (warping) por homografía
// Parámetros de entrada:
//   bMonoImg &rsc  -imagen fuente
//   fRgbImg &dst – imagen destino
//
// Procesamiento:
// La función utiliza homografía regresiva para rectificar un trapezoide de
//   la imagen fuente a la imagen destino.
// Esta versión utiliza una tabla de consulta de un índice de tres elementos para hacer una
//   referencia de componentes de colores en fotograma de vídeo en un formato raw de Bayer en lugar de
//   la búsqueda de píxeles bidimensional anterior en el fotograma RGB convertido
// La conversión de los píxeles de valores enteros a puntos flotantes
//   también tiene lugar aquí
//
//  Valor de retorno:
// ninguno.
//
// Escrito por: David Horton, Siemens Energy & Automation
//                          Intelligent Traffic Systems, Austin, Texas
//                          2/14/2007 & 7/13/2007
//-------------------------------------------------------------------------------------
void Clane:: RectifyBayer(bMonoImg &src, fRgbImg &dst)
{
                        if (dmLookupTable == NULL) {
                            printf("Create new zone – [%d, %d] [%d, %d] [%d, %d] [%d, %d]\n", m_TopLeft.x,
m_TopLeft.y,
                                    m_BotLeft.x, m_Botleft.y, m_BotRight.x, m_BotRight.y, m_TopRight.x,
m_TopRight.y);
                            int ret = Set(m_TopLeft, m_BotLeft, m_BotRight, m_TopRight, 0,0);
                            printf ('CLane::Set returned %d\n", ret);
                        // esto crea una nueva región definida para el colorid actual
                            assert(ret > = 0);
                           }
                            unsigned int size = dst.GetHeight() * dst.GetWidth();
                        // esto es más rápido que tamaño = dst.GetSize() porque GetSize() llama SetOrigin()
                            unsigned int I;
                            const unsigned char* k = src.GetData();
    #ifndef R
    #define R    0
```

```
#define clane_changed_flags  1
#endif
#ifndef  G
#define G   1
#define clane_changed_flags  1
#endif
#ifndef  B
#define B    2
#define clane_changed_flags  1
#endif
// Extraer información de componente de color
                    // correcto de imagen raw en mosaico con formato Bayer
                    // R, G y B pueden estar en cualquier orden para la operación correcta
                    // y el orden existente de ambos se selecciona para
                    // optimizar la operación de cache r&w
                    // código fuente retenido aquí en caso de que estos cambiasen
                    for (i= 0; i< size ; i++) {
                        dst.Pix(i)[R] = (float) *(k + dmLookupTable[i].redidx);
                dst.Pix(i)[G] = (float) *(k + dmLookupTable[i].greenidx);
                dst.Pix(i)[B] = (float) *(k + dmLookupTable[i].blueidx);
                        }
            }
```

Se comprenderá que las realizaciones descritas con anterioridad se proporcionan sólo a modo de ejemplo, y que el alcance de la invención se limita sólo por el alcance de las reivindicaciones adjuntas.

REIVINDICACIONES

1. Método para detectar un objeto en una imagen, dicho método comprende:

a) capturar la imagen;

b) proporcionar datos de fotograma de vídeo de un solo color de la imagen en un primer formato que incluye datos de píxeles para una pluralidad de píxeles, los datos de píxeles de cada píxel de imagen consisten en datos de un solo color;

c) generar datos de píxeles de múltiples colores para cada píxel de un bloque de detección representativo de una parte de imagen que es un subconjunto de los datos de fotograma de vídeo de un solo color y que se asocia por homografía a dicho bloque de detección, en donde los datos de píxeles de múltiples colores de cada píxel del bloque de detección se generan a partir de datos de píxel de un solo color de píxeles de la parte de imagen identificados utilizando dicha asociación homográfica;

d) proporcionar los datos de píxeles de múltiples colores del bloque de detección generados a un algoritmo de detección de objetos adaptado para detectar un objeto dentro del bloque de detección.

2. Método conforme a la reivindicación 1, en donde el paso c) también comprende asociar cada píxel del bloque de detección con los datos de píxeles de múltiples píxeles que tienen datos de un solo color representativos de múltiples colores.

3. Método conforme a la reivindicación 1, en donde el paso c) también comprende asociar cada píxel del bloque de detección con los datos de píxeles de tres píxeles contiguos del subconjunto de datos de fotograma de vídeo.

4. Método conforme a la reivindicación 1, en donde el paso d) también comprende proporcionar el bloque de detección de modo tal que cada píxel incluye datos de color rojo, verde y azul.

5. Método conforme a la reivindicación 1, que además comprende:

e) proporcionar los datos de fotograma de vídeo en el primer formato a un dispositivo informático;

f) emplear el dispositivo informático para interpolar fotogramas de los datos de fotograma de vídeo en el primer formato para generar datos de fotograma de vídeo interpolados.

6. Método conforme a la reivindicación 1, en donde el paso a) también comprende emplear una cámara digital que emplea una lente de mosaico.

7. Sistema para la detección de un objeto en una imagen, dicho sistema comprende:

un sistema de cámara configurado para capturar una imagen y proporcionar datos de fotograma de vídeo de un solo color de la imagen en un primer formato que incluye datos de píxel para una pluralidad de píxeles, los datos de píxel consisten en datos de un solo color;

un circuito de procesamiento configurado para

generar datos de píxeles de múltiples colores para cada píxel de un bloque de detección representativo de una parte de imagen que es un subconjunto de los datos de fotograma de un solo color y que se asocia por homografía a dicho bloque de detección, en donde los datos de píxeles de múltiples colores de cada píxel del bloque de detección se generan a partir de datos de píxel de un solo color de píxeles de la parte de imagen identificados utilizando dicha asociación homográfica;

proporcionar los datos de píxeles de múltiples colores del bloque de detección generados a un algoritmo de detección de objetos adaptado para detectar un objeto dentro del bloque de detección.

8. Sistema conforme a la reivindicación 7, en donde el sistema de cámara también comprende una cámara CCD y una lente de mosaico.

9. Sistema conforme a la reivindicación 7, en donde el circuito de procesamiento también está configurado para relacionar cada píxel del bloque de detección con los datos de píxel de múltiples píxeles que tienen datos de un solo color representativos de múltiples colores.

10. Sistema conforme a la reivindicación 7, en donde el circuito de procesamiento también está configurado para relacionar cada píxel del bloque de detección con los datos de píxeles de tres píxeles contiguos del subconjunto de datos de fotograma de vídeo.

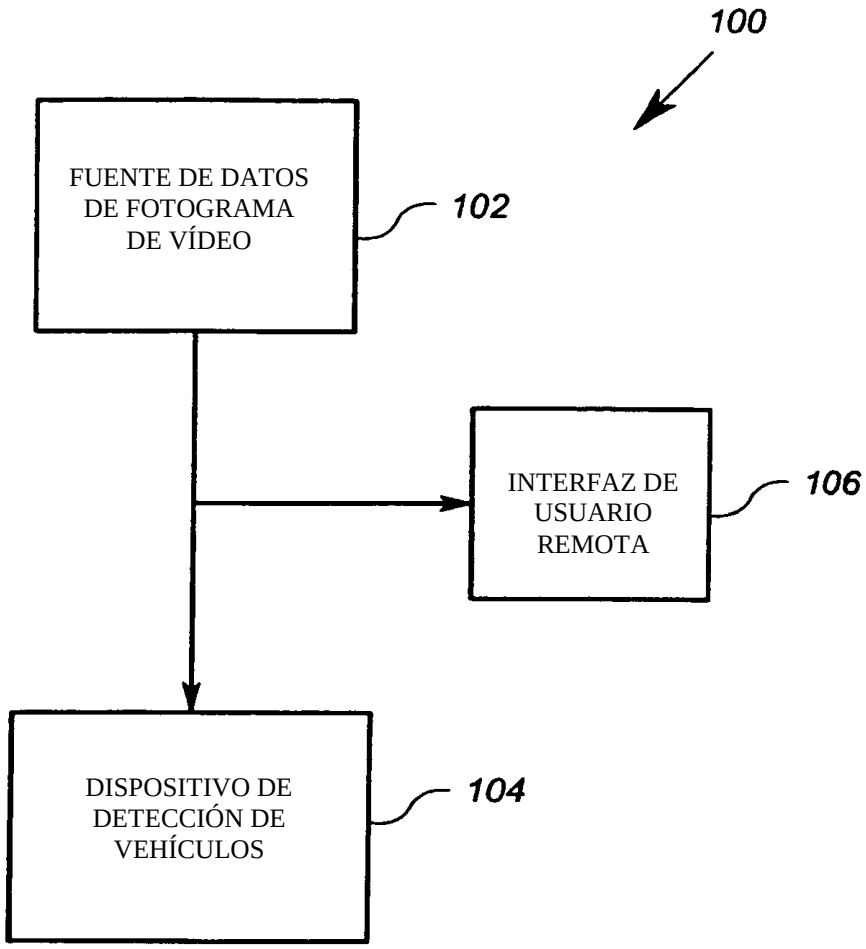
100
FUENTE DE DATOS DE FOTOGRAMA DE VÍDEO
102
INTERFAZ DE USUARIO REMOTA
106
DISPOSITIVO DE DETECCIÓN DE VEHÍCULOS
104

FIG. 1

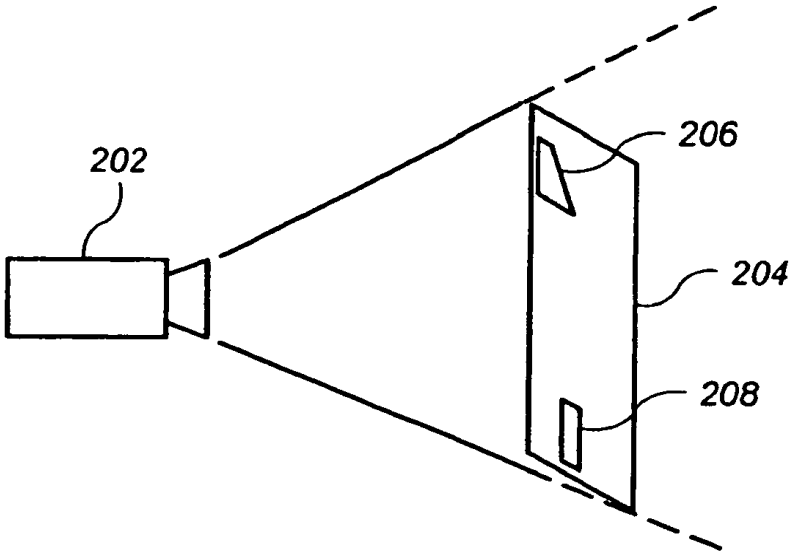

FIG. 2

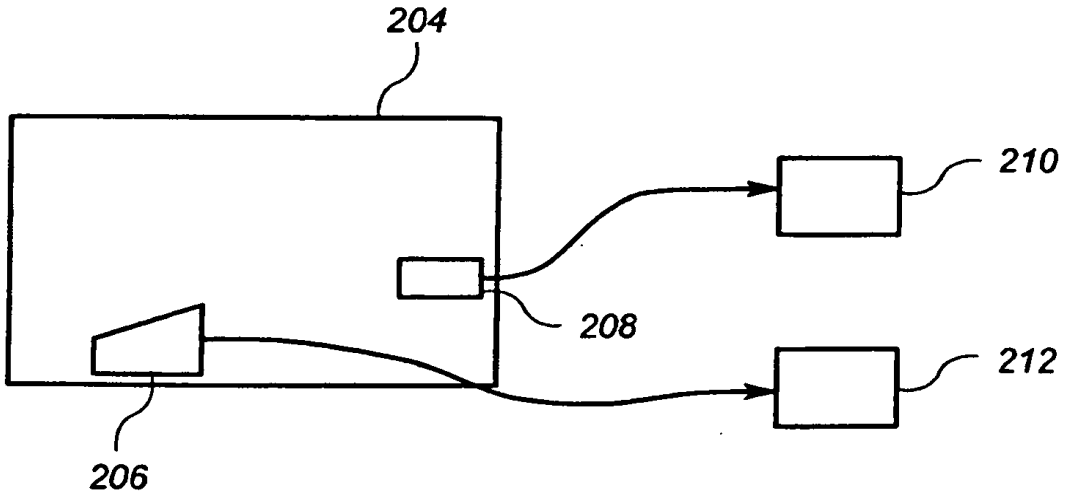

FIG. 3

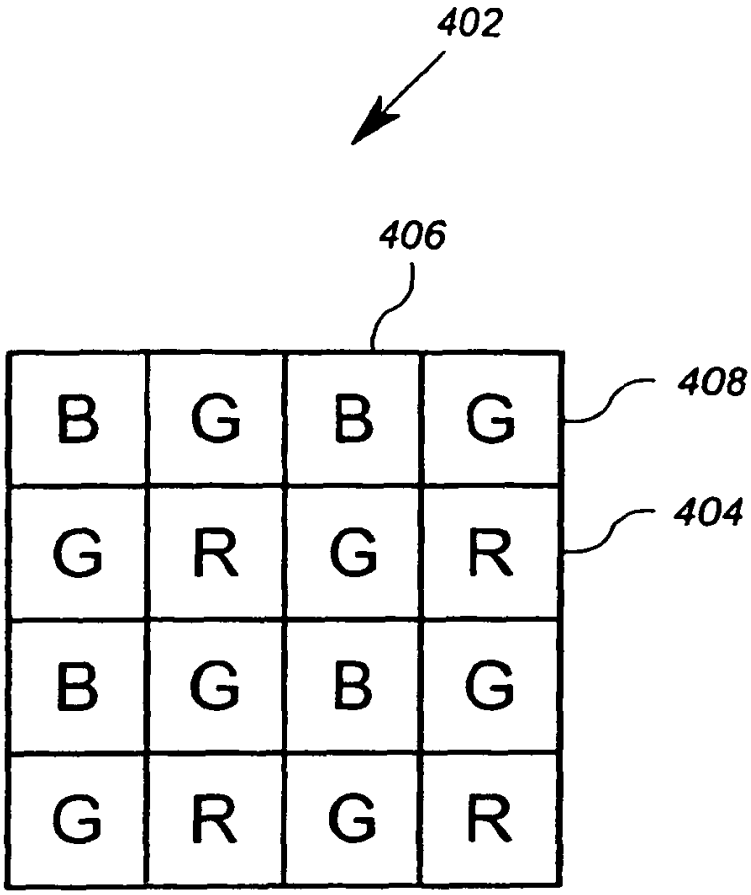

FIG. 4

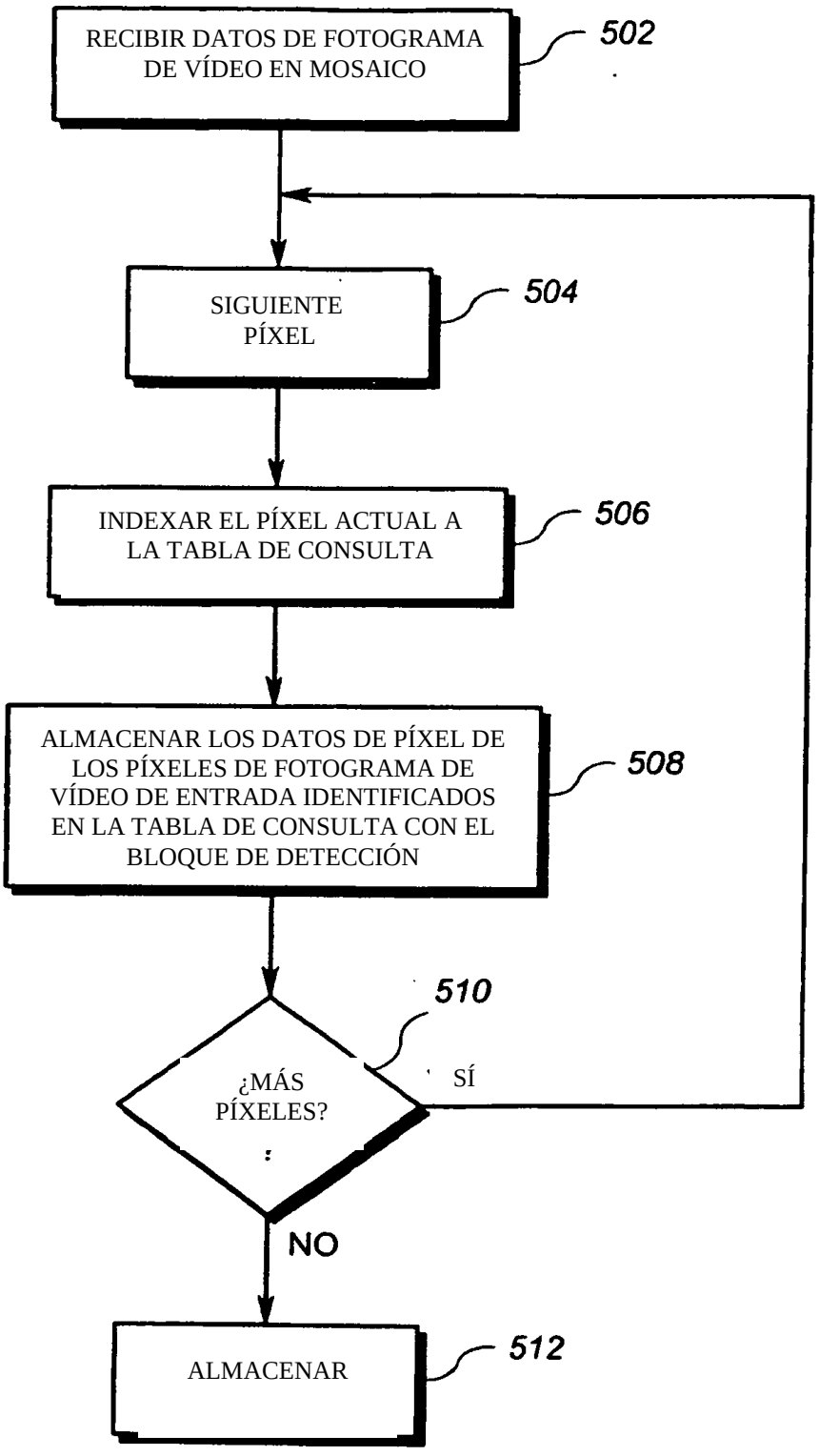
RECIBIR DATOS DE FOTOGRAMA DE VÍDEO EN MOSAICO
502
SIGUIENTE PÍXEL
504
INDEXAR EL PÍXEL ACTUAL A LA TABLA DE CONSULTA
506
ALMACENAR LOS DATOS DE PÍXEL DE LOS PÍXELES DE FOTOGRAMA DE VÍDEO DE ENTRADA IDENTIFICADOS EN LA TABLA DE CONSULTA CON EL BLOQUE DE DETECCIÓN
508
510
¿MÁS PÍXELES?
SÍ
NO
ALMACENAR
512

FIG. 5

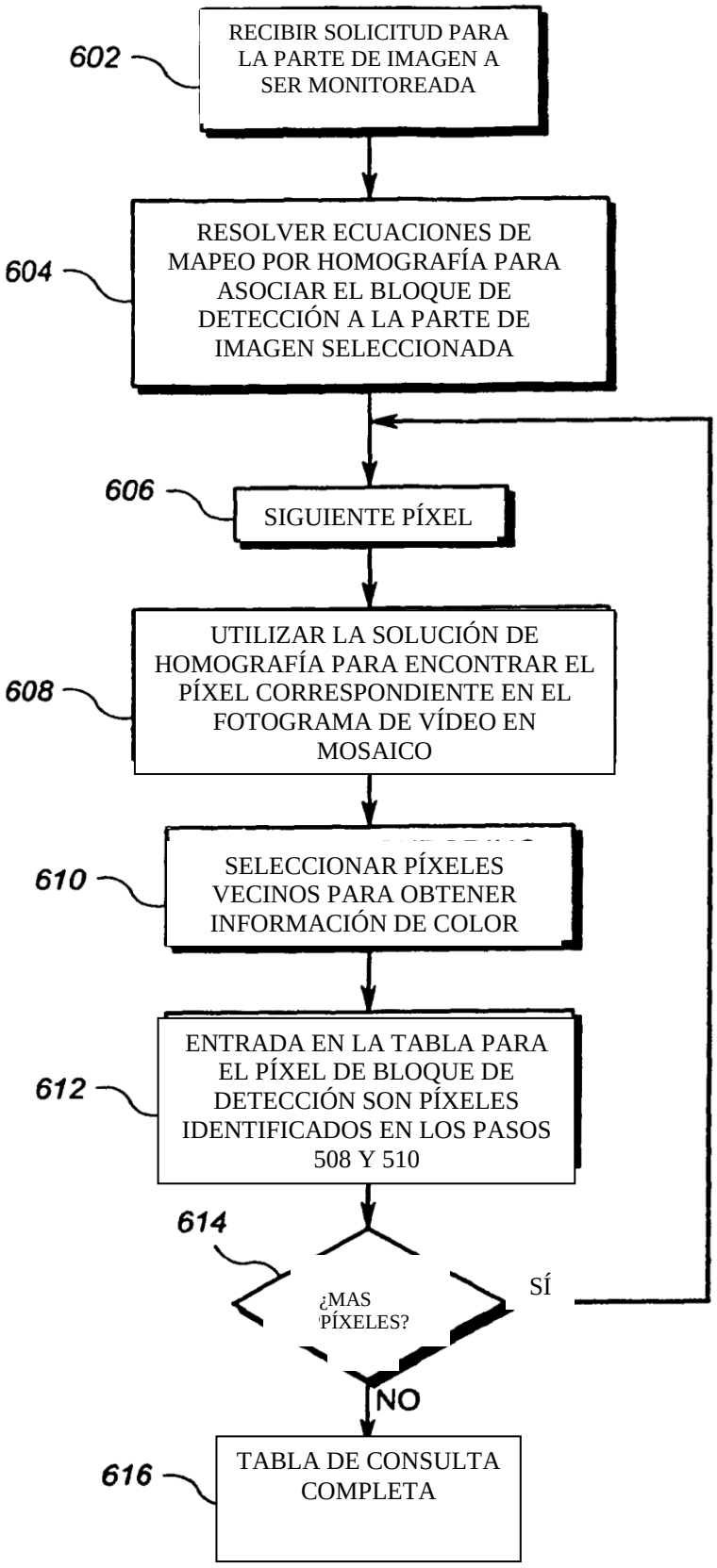
602
RECIBIR SOLICITUD PARA LA PARTE DE IMAGEN A SER MONITOREADA
604
RESOLVER ECUACIONES DE MAPEO POR HOMOGRAFÍA PARA ASOCIAR EL BLOQUE DE DETECCIÓN A LA PARTE DE IMAGEN SELECCIONADA
606
SIGUIENTE PÍXEL
608
UTILIZAR LA SOLUCIÓN DE HOMOGRAFÍA PARA ENCONTRAR EL PÍXEL CORRESPONDIENTE EN EL FOTOGRAMA DE VÍDEO EN MOSAICO
610
SELECCIONAR PÍXELES VECINOS PARA OBTENER INFORMACIÓN DE COLOR
612
ENTRADA EN LA TABLA PARA EL PÍXEL DE BLOQUE DE DETECCIÓN SON PÍXELES IDENTIFICADOS EN LOS PASOS 508 Y 510
614
¿MAS PÍXELES?
SÍ
NO
616
TABLA DE CONSULTA COMPLETA

FIG. 6

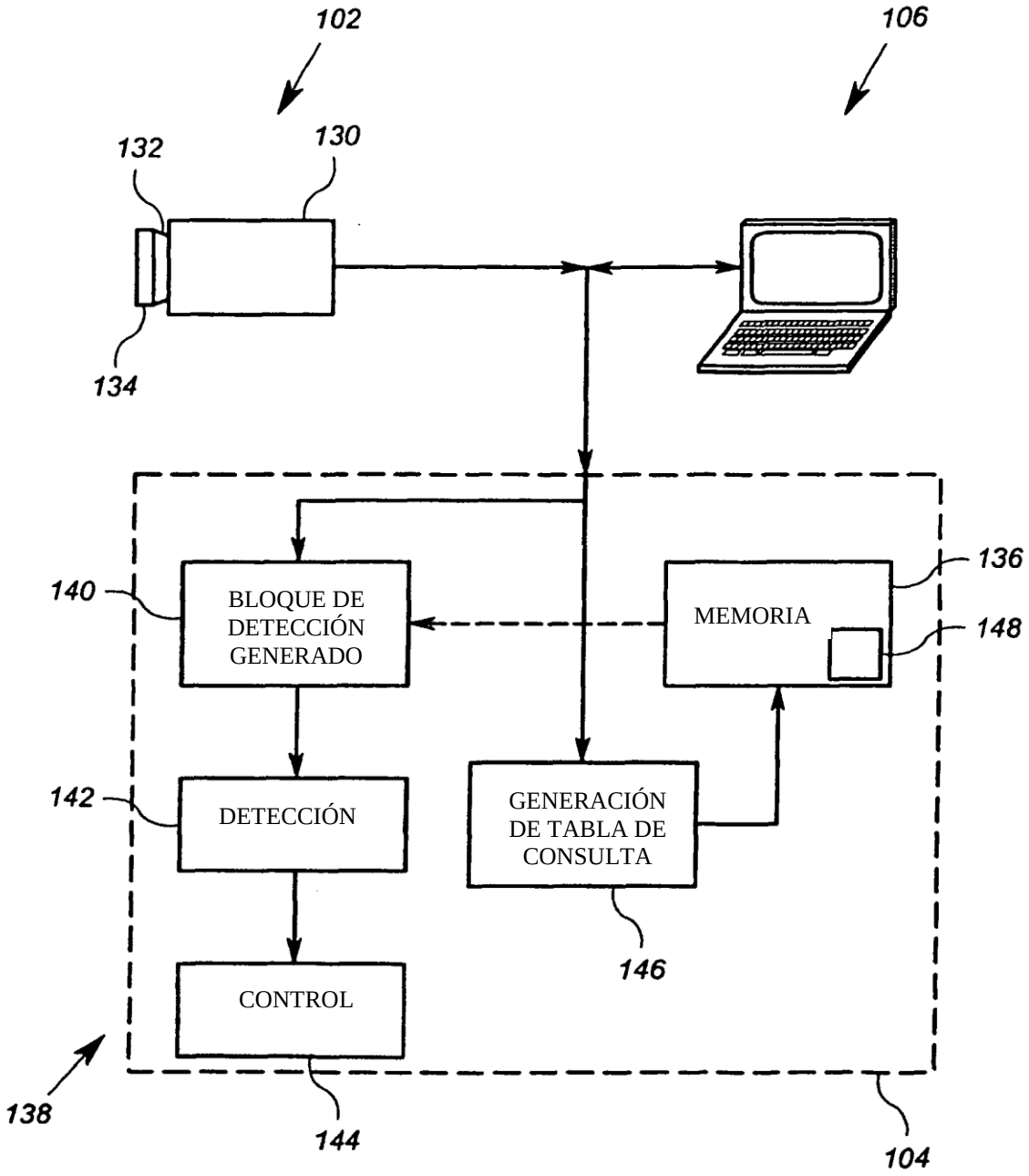
102
106
132
130
134
140
BLOQUE DE DETECCIÓN GENERADO
MEMORIA
136
148
142
DETECCIÓN
GENERACIÓN DE TABLA DE CONSULTA
146
CONTROL
138
144
104

FIG. 7