

Prototipo HW y SW de un Sistema de Digitalización de imágenes

Ramon Pla, Borja Martinez, Marius Monton y Jordi Carrabina
Centro de Prototipos y Soluciones HW
U. Microelectrònica, Dpt. Informàtica, ETSE
Universitat Autònoma de Barcelona

Resumen. El presente artículo describe la implementación en software y hardware de un sistema de digitalización de documentos heterogéneos con un sensor CMOS conectado a un PC y a una placa de prototipado, respectivamente. Específicos algoritmos de procesamiento de imágenes han sido desarrollados para detectar los cambios en las escenas de las imágenes. Procesando dichos cambios, el sistema puede detectar movimiento y la presencia de un documento en la imagen capturada por el sensor, y así producir la orden necesaria para su almacenamiento.

1 Introducción

En este artículo se muestra como se han implementado dos prototipos de un mismo sistema: un prototipo SW y un prototipo HW. El primero, el prototipo SW, se ha implementado con un sensor óptico conectado a un PC (entorno SW) a través de un bus USB. El prototipo HW se ha implementado con un sensor óptico conectado directamente a una FPGA en una plataforma con conexión *Ethernet* (entorno HW). Dicho prototipo HW es una modificación del sistema comercial DCAM1 [8], al que se le ha añadido la mejora de trabajar solo con determinadas regiones de la imagen capturada por el sensor para permitir un procesamiento a tiempo real.

El prototipo SW se ha implementado con el lenguaje y entorno de programación *Visual C++ 6* e integrando las librerías de procesamiento de imágenes (*IPL*) de Intel y las librerías *OpenCV* (*Open Source Computer Vision Libraries*). La *IPL* está compuesta por primitivas de procesamiento de imágenes. Estas funciones de bajo nivel están optimizadas para los procesadores de Intel, especialmente para los de tecnología *MMX*. La *IPL* esta enfocada en aprovechar el paralelismo de las instrucciones *SIMD* (*single-instruction, multiple-data*) de las últimas generaciones de procesadores de Intel [1]. *OpenCV* es una librería de algoritmos de implementación a alto nivel. Está diseñada para ser usada juntamente con la *IPL* de Intel y extiende su funcionalidad a través de análisis de patrones e imágenes.

El prototipo HW se ha implementado en *VHDL* usando el entorno de herramientas de desarrollo HW *Quartus 2.1* de Altera, y bajando el diseño a una *Apex 20k200* también de Altera.

2 Descripción del sistema

El algoritmo utilizado para el sistema de digitalización se basa en el cálculo de diferencias entre imágenes. De manera que cualquier detección de movimiento es detectado con la diferenciación entre la imagen actual y la anterior (entre el frame n y el frame $n-1$). Y de forma similar, un documento es detectado al realizar la diferenciación entre el frame actual y el del fondo. El frame del fondo es una imagen, guardada en memoria, que contiene la escena sin documento. Es decir, una imagen de referencia capturada al inicio de la aplicación.

Si después de una detección de movimiento no hay ninguna otra detección de movimiento durante un intervalo de tiempo determinado (por ejemplo un segundo) la imagen del documento es capturada y guardada.

El método de diferenciación entre imágenes es simple y rápido de procesar, pero presenta el inconveniente del ruido debido a los cambios de luz [3]. Por esta razón, se aplican dos thresholds o bias (valor umbral), uno para la detección de movimiento y el otro para la detección de documento. Consecuentemente, la imagen de diferencias es comparada con el bias para proceder a la captura de una imagen. El valor del bias del movimiento determina la facilidad para detectar un movimiento. Si este valor es bajo, el sistema será más sensible a los pequeños movimientos y a los cambios de iluminación, y si es alto, no será capaz de detectar movimientos lentos ni cambios de luz. De forma similar, el valor del bias del documento determina la presencia o no de un documento en la escena. Si el valor del bias es muy bajo no se detectarán documentos con colores similares que el fondo, y si por el contrario el bias es alto, el documento y el fondo tendrán que ser lo suficientemente diferentes para que se determine la presencia de un documento (Ej. uno tendrá que ser de un color oscuro y el otro de un color claro). Cuando se inicia la aplicación, se guarda la imagen captada del fondo de la escena (sin documento) y se inicializan los valores de los bias con el valor medio de los píxeles de la imagen. No obstante, a pesar de esta inicialización de los bias, el valor de estos se pueden modificar durante la ejecución del programa dependiendo de cual sea su comportamiento (Ej., si el sistema es demasiado sensible a la iluminación o no se detecta un documento oscuro).

El sensor óptico de nuestro sistema tiene una resolución de 640x480 píxeles. Esta resolución produce imágenes de 307.200 píxeles. Si trabajamos con imágenes de color RGB, con una profundidad de color de 24 bits por píxel, obtenemos imágenes de 7.372.800 bits. Esta cantidad de información es demasiado elevada para implementar eficientemente la diferenciación entre imágenes en una FPGA, por la cantidad limitada de recursos, para una “implementación paralela” (soluciones iterativas requerirán planificaciones de memoria más complicadas). Por lo tanto, hemos introducido un criterio de optimización para permitir una implementación del cómputo paralelo de la diferenciación entre imágenes en tiempo real. Hemos seleccionado de cada imagen 8 regiones de 8x8 píxeles, y hemos aplicado los algoritmos de detección de cambios a cada una de ellas. Trabajando con 8 regiones de 8x8 reducimos el número de píxeles de 307.200 a 512 píxeles, y trabajando con el mismo formato de color, de 24 bits por píxel, obtenemos solo un tamaño de información de 12.288 bits. Además, los requerimientos de almacenamiento se reducen mucho permitiendo un procesamiento continuo con menos recursos.

Considerando estos números como genéricos (regiones, píxeles,...), el sistema de digitalización de documentos heterogéneos es un sistema flexible. Su funcionalidad depende del valor de los bias que el usuario puede indicar y de la distribución y medida de las regiones a procesar. En nuestra implementación, dicha distribución de las regiones se basa en el tipo de documentos a digitalizar (formato Din-A4), de modo que se ha escogido una distribución de doble diagonal (como una letra X).

3 Prototipo SW

El prototipo SW implementado ha sido orientado a una posterior síntesis en HW. De este modo hemos introducido el criterio de mejora de trabajar con “regiones”. Esta mejora se ha aplicado tanto a la detección de movimiento como a la detección de documento. En ambos casos hemos seleccionado 8 regiones de interés o *ROIs (Regions Of Interest)*. Las ROIs que seleccionamos en la imagen de referencia tienen las mismas dimensiones y coordenadas (x, y) que la imagen con la que se realiza la substracción o diferenciación. A la imagen capturada por el sensor se le indican las ROIs y, posteriormente, a cada una de estas se le aplican los algoritmos de detección de movimiento y de documento. Así, a cada imagen o frame capturado por el sensor, aplicamos los algoritmos de detección tantas veces como ROIs hayan indicadas. En nuestro caso hemos seleccionado 8 regiones, así que ejecutamos el mismo algoritmo 8 veces, una vez a cada región. Por cada píxel diferente que detectamos en una ROI respecto a la ROI de su imagen anterior o del fondo, incrementaremos un contador. Por lo tanto tenemos dos contadores para cada ROI, uno para la detección de movimiento y el otro para la detección de documento. Tanto en la detección de movimiento como en la detección de documento, cuando ya se han analizado todas las regiones, comprobamos el valor de estos contadores para determinar si las dos imágenes son iguales o son diferentes entre si. Al trabajar con ROIs, la inicialización de los bias se realiza con el valor medio de los píxeles de todas las regiones.

3.1 Detección de movimiento

Como ya se ha mencionado anteriormente, la detección de movimiento se basa en la diferenciación entre la imagen actual y la imagen anterior, entre la imagen n y la imagen n-1. Al trabajar con imágenes en color, antes de realizar la diferenciación convertimos las imágenes de color (RGB, 3 canales) a imágenes en escala de grises (1 canal). Posteriormente, obtenemos la diferencia absoluta entre las dos imágenes. Esta diferencia se realiza píxel a píxel, de modo que el valor del píxel con coordenadas (x',y') en una imagen es substraído del valor del píxel con las mismas coordenadas (x', y') de la otra imagen.

En teoría, si no hubiera movimiento, el resultado de la diferenciación sería cero. No obstante, tenemos que considerar la presencia de ruido en las imágenes [3]. Consecuentemente aplicamos al valor de los píxeles de la imagen resultado un *threshold* o *bias* con el que se compara el valor de los píxeles de la imagen resultado. Según cual sea el valor de esta comparación, el valor del píxel se actualizará a 255 (color blanco) o a 0 (color negro). Por lo tanto

obtendremos una imagen binaria que solo tendrá píxeles blancos y negros, donde los píxeles blancos son los píxeles diferentes respecto a la imagen anterior y los píxeles negros son los píxeles iguales. No obstante, esta binarización de la imagen resultado no nos permite detectar pequeños movimientos (movimientos lentos en la escena). Pero esto no afecta a la funcionalidad de nuestro sistema porque este efecto es inapreciable cuando se detecta el giro de la página de un libro (este giro raramente será lento).

El último paso en la detección de movimiento es determinar cuando hay movimiento y cuando no. Esto significa contar el número de píxeles blancos en las regiones seleccionadas. Entonces, si hay uno o más píxeles blancos, significa que las dos imágenes, la actual y la anterior, son diferentes.

3.2 Detección de documento

La detección de documento es similar a la detección de movimiento. Se ha implementado usando el mismo algoritmo con las mismas instrucciones. La única diferencia es que en este caso la diferenciación se hace entre la imagen actual y la imagen del fondo, la cual se guarda cuando se inicia la aplicación. Ahora, dependiendo del valor del bias, la binarización aplicada a la imagen resultado, obtenida con la diferenciación entre la imagen actual y la imagen del fondo, no permitirá detectar documentos con el mismo color que el fondo de la escena (Ej., no podremos detectar un documento blanco en un fondo blanco).

Como en la detección de movimiento, los píxeles de la imagen resultado serán blancos si son diferentes del fondo, y negros si son iguales.

4 Prototipo HW

El prototipo HW, mostrado en la Fig.1, se ha implementado en una placa PAC [7] y una placa de sensor [8], [9]. El sensor CMOS es una cámara de color OV7620 Single-chip. La placa del sensor incluye una SRAM de 4Mbit donde podemos almacenar la imagen capturada por el sensor para su procesamiento, y eventualmente transmitirla a través de Ethernet. El sensor está configurado en espacio de colores YUV. Por lo tanto lo usamos como una cámara en blanco y negro seleccionando la salida del puerto Y y descartando la información del puerto UV.

En el prototipo HW, para seleccionar las ROIs de la imagen utilizamos las señales *VSYNC* y *HREF* del sensor, las cuales indican cuando empieza una imagen y una línea de la imagen en cuestión. De modo que, a diferencia del prototipo SW, en el prototipo HW las ROIs no se indican a partir de una imagen completa guardada en memoria, sino que, estas se calculan a medida que el sensor transmite los píxeles de la imagen al sistema. Por ejemplo, si la resolución de nuestro sensor es de 640x480 píxeles, por cada señal *HREF* activa procesamos 640 píxeles de la imagen, que es la longitud de una línea de imagen.

Para determinar si un píxel pertenece a una ROI, hemos implementado una maquina de estados con las señales VSYNC y HREF y dos contadores, *counterX* y *counterY*. CounterX cuenta el número de píxeles de columna del frame, o coordenada X, y counterY cuenta el número de píxeles de fila del frame, o coordenadas Y. Con estos dos contadores sabemos las coordenadas de cada píxel que se procesa. De modo que comparamos el valor de los contadores con las coordenadas de cada ROI. Entonces, si el valor del contador pertenece a alguna ROI, se aplica el algoritmo de diferenciación.

Además de dichos contadores, también se utilizan dos memorias FIFO, una para guardar la imagen anterior y la otra para guardar la imagen del fondo. La FIFO del fondo guarda la misma imagen durante toda la ejecución de la aplicación. Esta imagen del fondo es guardada al inicio de la aplicación, de modo que mantiene el mismo valor asignando la salida de la FIFO a su entrada. Es decir, la entrada a la FIFO del fondo es la misma salida de ésta, excepto cuando se inicializa. En tal caso la entrada de la FIFO es la información procedente del sensor.

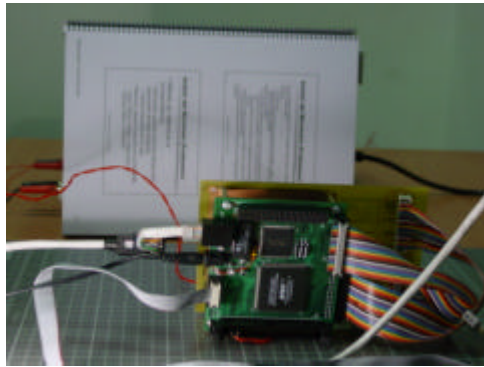


Fig.1. Placa prototipo HW en frente de un documento

Por el contrario, la FIFO de la imagen anterior es actualizada con la imagen actual, es decir, por cada frame transmitido por el sensor. En tal caso, la salida de la FIFO es diferente de su entrada, que es la información del sensor.

Consecuentemente, para controlar las señales de las FIFOs, hemos implementado otra máquina de estados para poder controlar los diferentes retardos de lectura de estas memorias. En la FIFO del fondo solo la primera vez que se realiza su lectura el retardo es de dos ciclos de reloj, y en las siguientes lecturas el retardo es de solo un ciclo de reloj.

En ambos casos, tanto en la FIFO del fondo como de la imagen anterior, la salida es la entrada del modulo que implementa la diferencia entre ROIs. En este modulo se realiza la diferenciación entre la información de la FIFO y la información procedente del sensor.

En el prototipo HW, como en el prototipo SW, la implementación de la detección de movimiento es similar a la detección de documento. La única diferencia es que en el primer caso,

en la detección de movimiento tenemos como entrada la salida de la FIFO de la imagen anterior y en el segundo caso, en la detección de documento, se toma la salida de la FIFO del fondo como entrada del modulo de diferenciación.

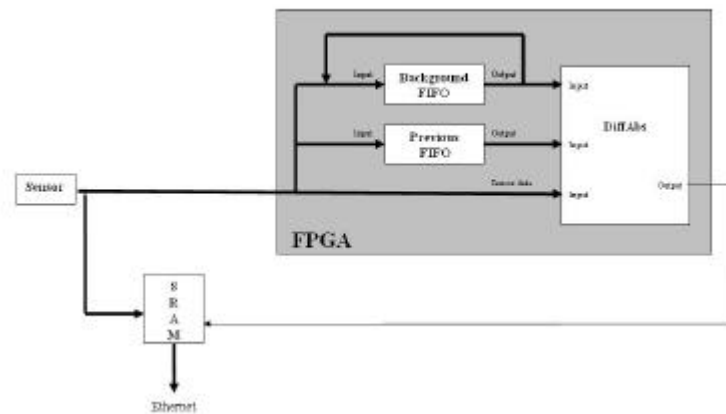


Fig.2. Diagrama de bloques del prototipo HW

Como en el prototipo SW, la inicialización de los bias se realiza a partir del valor medio de los píxeles de las ROIs. Cuando el píxel del sensor pertenece a una ROI, se realiza la substracción correspondiente con la imagen anterior o del fondo y se acumula el resultado de esta. Luego, cuando se detecta el fin de la transmisión de un frame, se divide el valor acumulado por 512, que es el número total de píxeles de las ocho regiones de 8x8 píxeles. Ya que el número 512 es potencia de 2 ($512 = 2^9$) solo tenemos que desplazar el valor del registro 9 bits a la derecha. Si hubiéramos seleccionado otro número diferente de regiones que hubiera producido un total de píxeles de las ROIs que no fuera potencia de 2, hubiéramos tenido que implementar un modulo de división en vez del registro de desplazamiento.

La detección de movimiento y de documento se realiza con contadores. Para cada tipo de detección tenemos diferentes contadores para cada región, en nuestro caso 8 contadores diferentes, y un contador global que cuenta el número de regiones que son diferentes de las regiones de su imagen correspondiente, la anterior o la del fondo. Entonces, si el píxel del sensor pertenece a alguna de las 8 regiones indicadas, se realiza la substracción absoluta entre la información del sensor y la información de la FIFO. El resultado de esta diferenciación se compara con el bias, y si este es mayor que el valor del bias, incrementamos el contador de la región respectiva. De modo que, para cada región, el valor del contador es el número de píxeles diferentes.

Eventualmente, cuando finaliza la transmisión de un frame en proceso, comprobamos el valor del contador de cada región. Para cada región, si el valor del contador es mayor que cero, incrementamos el contador global de regiones. Entonces, si el valor del contador global es mayor que cero, significa que la imagen es diferente de la imagen anterior o de la ima-

gen del fondo (dependiendo del tipo de detección que se realice) porque hay, al menos, una región diferente.

5 Resultados y conclusiones

Se ha implementado un prototipo SW y un prototipo HW de un sistema de digitalización de documentos heterogéneos. En el prototipo SW hemos tomado decisiones de diseño orientadas para una eficiente implementación en HW, y en el prototipo HW hemos implementado algoritmos y funciones de procesamiento de imágenes previamente validadas en SW.

Primero hemos implementado un prototipo SW y comprobado su funcionalidad. Luego, después de validar su comportamiento, hemos implementado el mismo sistema en HW. Esta segunda implementación en HW realiza un procesamiento de las imágenes a bajo nivel, a diferencia del prototipo SW que lo realiza en un alto nivel de abstracción e implementación. Ya que se ha realizado primero un prototipo SW a alto nivel, donde siempre es más fácil de trabajar gracias a las diferentes herramientas de depuración, hemos optimizado el diseño de la implementación HW porque ya sabíamos exactamente que teníamos que hacer.

Resultados obtenidos del prototipo SW muestran que se necesitan 180 ms de procesamiento y una latencia de 900 ms para cada gestión de lectura o escritura de las imágenes guardadas en memoria cuando trabajamos con imágenes de 640x480 píxeles y profundidad de color de 24 bits por píxel. Cuando el algoritmo SW utiliza las ROIs, las latencias de procesamiento fueron altamente reducidas (factor 4), a pesar que la gestión de memoria presentaba requerimientos temporales similares ya que las imágenes de la cámara se guardaban completas. Las consideraciones temporales limitan el número de imágenes a procesar en un PC o procesador multitarea, ocasionando una importante retardación de éste.



Fig.3. Aplicación SW de la captura de imágenes

La implementación HW ha sido una *traducción* de las funciones de procesamiento de imágenes IPL/OpenCV a un programa VHDL. Hemos observado que podemos obtener el mismo funcionamiento controlando las señales del sensor VSYNC y HREF, e implementando el nombre de contadores requeridos (dependiendo del número de regiones de la imagen).

En el prototipo HW, con la mejora de trabajar solo con regiones obtenemos la importante ventaja que no es necesario utilizar memoria externa para guardar los píxeles a procesar, ya que estos caben en la memoria interna de la fpga, evitando retardos de acceso a memoria y así conseguir un procesamiento a tiempo real de las imágenes captadas por el sensor.

A pesar de que la dificultad de las funciones de procesamiento de imágenes no era muy alta (básicamente substracciones) hemos visto las ventajas del diseño SW/HW. La implementación resultante, incluyendo el control del sensor, protocolo UDP y una interficie específica de Ethernet, tiene una ocupación del 66% de los elementos lógicos y un 46% de memoria de una EP20K200. En este caso la latencia no es importante pues estamos procesando automáticamente 25 imágenes por segundo, dadas por el sensor, y transmitiendo solo la que se captura del documento en cuestión.

Hemos podido ver que con el prototipo HW se puede obtener la misma funcionalidad de procesamiento de imágenes que con las funciones IPL de Intel y OpenCV usadas en el prototipo SW.

Referencias

1. "Intel Image Processing Library, Reference manual", pp.1-1, 1997-2000.
2. "Open Source Computer Vision Library, Reference manual", pp.1-1, 1999-2001.
3. A. De la Escalera, "Visión por computador, Fundamentos y Métodos", Prentice Hall, pp.178-187, 2001.
4. <http://www.intel.com/research/mrl/research/opencv>
5. <http://www.intel.com/research/mrl/research/cvlib>
6. Al Bovik, "Hand Book of Image Processing", Academic Press, 2000.
7. PAC Technical Report. Histeresys S.L. (NDA required) <http://www.histeresys.com>
8. DCAM1. Technical Report. Histeresys S.L (NDA required). <http://www.histeresys.com>
9. O. Ferraz, et al. "Flexicamera: Intelligent and Flexible Camera". DCIS 2001. November 20-23, 2001.