Desarrollo de un Prototipo de Cámara Inteligente

Nelson ACOSTA y Silvia IARRAR
INTIA/INCA – Dpto. Computación y Sistemas – Facultad de Ciencias Exactas
UNCPBA - (7000) TANDIL – Argentina
Email: {nacosta, siarrar}@exa.unicen.edu.ar

Palabras clave: reconocimiento de patrones, adquisición de imágenes, IQ cámaras

Resumen

Este artículo presenta un prototipo de procesador a medida para la identificación de patrones en tiempo real sobre una plataforma FPGAs (Field-Programmable Gate Arrays). Se proponen dos dispositivos orientados a la velocidad y aplicabilidad en entornos industriales.

1 Introducción

Debido a la importancia del sentido de la vista humana era de esperar que se investigara la visión artificial en las computadoras. La máquina realiza el análisis de imágenes con el propósito de obtener una descripción de los objetos físicos captados por la cámara. Los primeros trabajos datan de la década del cincuenta, donde aún se creía una tarea fácil y alcanzable en pocos años. Roberts [Rob65] demostró la posibilidad de procesar una imagen digitalizada para obtener una descripción matemática de los objetos incluidos en la escena. Wichman [Esc01] presentó un equipo con cámara de televisión conectada a un computador para identificar objetos y sus posiciones en tiempo real. Sin embargo se trataba de imágenes muy simples, con fuertes restricciones. Los años siguientes se desarrollaron algoritmos utilizados aún hoy, como los detectores de bordes de Roberts [Rob65], Sobel [Esc01] y Prewitt [Pre70].

En los ochenta, debido al desarrollo de tecnologías más económicas y con mayor capacidad de cálculo, la investigación en visión artificial se enfoca a la realidad. Se realizan los primeros circuitos específicos para el procesamiento y tratamiento de imágenes [Gon93, Esc01]. Con ello empiezan a ser utilizables aplicaciones cuyo tiempo de cálculo las hacía inviables o cuyo precio era prohibitivo.

La visión por computadora puede clasificarse en dos grandes categorías: A) *Visión a bajo nivel*, que busca el conjunto de las características que definen el problema a resolver: diferentes zonas de color, presencia o no de objetos, movimiento, texturas, forma de los objetos. B) *Visión de alto nivel* o *análisis de imágenes*, utiliza dichas características para obtener descripciones más abstractas: análisis de las formas, reconocimiento y localización de objetos.

Una imagen óptica es convertida a señales eléctricas por una cámara, cambiando la representación de luz óptica en una señal eléctrica. Las cámaras pueden clasificarse según su grado de asistencia al usuario en: A) Cámara "*boba*" es la que sirve sólo para obtener una imagen, aún cuando dicha imagen tenga procesamientos destinados a mejorar su calidad. B) Cámara "*inteligente*" es la que realiza la obtención y análisis de la imagen, generando las características procesadas de la información; cuyas ventajas son la velocidad, tamaño, consumo de potencia, integración y modularidad.

La velocidad se logra evitando los cuellos de botella, entre la captura y las etapas de procesamiento, mediante la utilización masiva de paralelismo. Por otra parte, si se implementa un algoritmo de visión complejo en un solo circuito integrado, se reduce la circuiteria y la energía necesaria para la transmisión de información; provocando una reducción del consumo de potencia y del tamaño del sistema. La alta integración permite liberar a la CPU de la ejecución de tareas muy costosas.

En ciertas aplicaciones de video o imagen es necesaria una capacidad elevada de procesamiento y tasa de transferencia de datos, para atender requisitos de rapidez

impuestos por la aplicación. Ciertas tareas de procesamiento de imagen pueden hacerse recurriendo a unidades de procesamiento dedicadas que operen directamente sobre la información capturada de la imagen 2D. De esta manera se realiza en tiempo real cierto post procesamiento de la imagen, sin la necesidad de su transformación a formatos de video o imagen. Otras aplicaciones sólo necesitan procesar una región de la imagen. Algunas no necesitan almacenar la región completa sino el proceso píxel a píxel de dicha área para obtener la información.

Los FPGAs [Bro94, Mey01] son una alternativa atractiva para implementar aplicaciones de tratamiento de imagen debido a que son dispositivos flexibles dando la posibilidad de ajustar el hardware a las necesidades individuales de la aplicación. Muchas aplicaciones son candidatas para implementarse en FPGAs, incluyendo filtros y algoritmos de edición, calibración, compresión, reconocimiento, análisis, etc. Los FPGAs por ser dispositivos programables, requieren de un software de apoyo que permita: describir el circuito a implementar (VHDL o Verilog), simular y finalmente programar el circuito físico.

2 Objetivo

El objetivo del presente trabajo es el Diseño y Síntesis de un *Procesador Dedicado para la Identificación de Patrones en Imágenes*. La materialización permite controlar el dispositivo que origina las señales, cámara de video, y que además identifique ciertos patrones (restringidos en esta etapa del proyecto) en las imágenes. El trabajo es dividido en dos etapas, la primera es el driver encargado de la captura de la imagen, y una segunda etapa de alto nivel que realiza el análisis de imágenes.

3 Etapa de visión a bajo nivel: “Driver de la cámara”

El primer prototipo se desarrolla realizando ingeniería inversa al driver código abierto de la cámara propuesto por [Rey00]. Basándose en este código se desarrolla un modelo del comportamiento de la cámara, realizándose un prototipo software que maneje la cámara obteniendo frames a pedido. Con este prototipo se modela el comportamiento de la cámara en alto nivel, como se muestra en el diagrama de la Fig. 1.

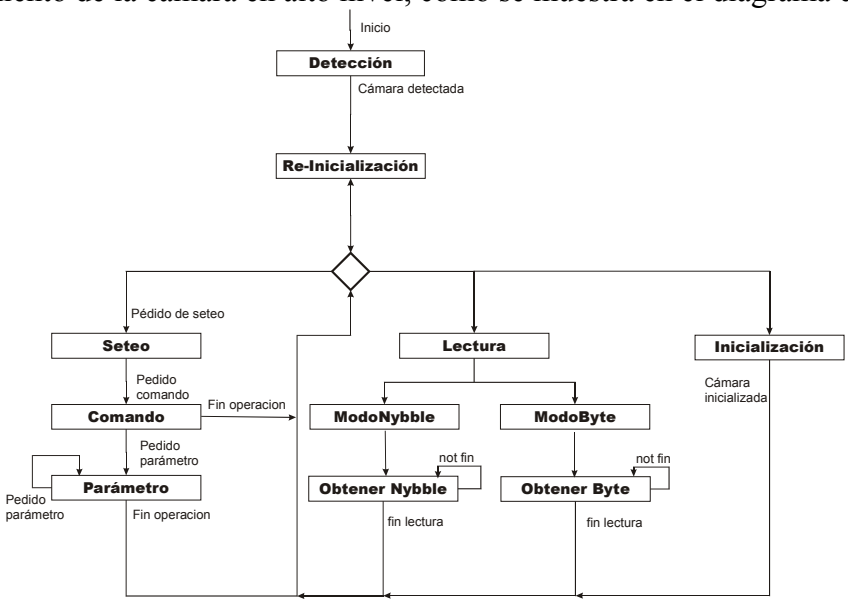

Fig. 1 – Diagrama del driver de la cámara

Protocolo de comunicación

El protocolo *handshaking* de comunicación es utilizado para todas las operaciones. Se utilizan dos señales, una de entrada y una de salida. Para realizar cualquier operación con la cámara, la señal de entrada cambia su estado lógico a alto y envía un código de operación por sus señales de dato. La cámara nota este cambio y levanta su señal para indicar que recibió el pedido (ACK), y manda la primera parte del eco del pedido de operación. El *host* nota este cambio y se lo indica a la cámara bajando su señal. La cámara envía la segunda parte del pedido de operación y baja su señal, quedando todas las señales en el estado inicial. Este protocolo se muestra en el diagrama (Fig. 2) de señales en el tiempo, que incluye manejo de condición de error. Una variante de este handshaking se utiliza para pedir comandos con lectura de parámetros, donde no se envía nada por las señales de datos y los ecos son los parámetros solicitados.

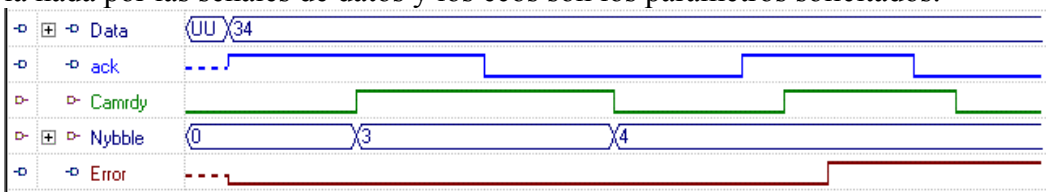

Fig. 2 – Diagrama de señales en el tiempo

Formato de transferencia de la imagen

Como la cámara trabaja en dos modos, realiza la lectura de cada píxel de dos maneras diferentes, en el modo Nybble, por cada handshaking se leen 4 bits, y en el modo Byte por cada handshaking se leen los 4 bits del Nybble, más los 8 bits de dato.

Cada píxel representa un RGB de 24 bits, por lo tanto en modo bits, se necesitan 6 lecturas de Nybbles y en el modo Byte sólo 2. Otra diferencia entre los dos modos es la forma en que se construye el RGB. Para construir el RGB en modo Nybble, se realizan 6 lecturas, obteniendo 6 Nybbles a los cuales, hay que invertir el bit más significativo de cada uno de ellos y la forma en que la cámara envía los pixeles es RGBRGB. En la primer Tabla se muestra como se construye en modo Nybble, los bits que se encuentran en negrita y cursiva se encuentran invertidos.

R								G								B							
Nybble1				Nybble2				Nybble3				Nybble3				Nybble3				Nybble3			
N3	N2	N1	N0	***N3***	N2	N1	N0	***N3***	N2	N1	N0	***N3***	N2	N1	N0	***N3***	N2	N1	N0	***N3***	N2	N1	N0

Para construir el RGB en modo Byte, se realizan 2 lecturas, obteniendo en cada una de ellas 8 bits de Datos y 1 Nybbles de 4 bits, a los cuales, hay que invertir el bit más significativo. La forma en que la cámara envía los píxeles es RGBRGB. En la segunda Tabla se muestra como se construye en modo Byte, los bits que se encuentran en negrita y cursiva se encuentran invertidos. Existe otro formato de píxel de 32 bits, donde envía los píxeles en R G1 G2 B R G1 G2 B, entonces se necesitan 8 lecturas en modo Nybble para formar 1 píxel o 8 lecturas en modo Byte para leer 3 píxeles.

R								G								B							
Dato1								Nybble1				Nybble2				Dato2							
D7	D6	D5	D4	D3	D2	D1	D0	***N3***	N2	N1	N0	***N3***	N2	N1	N0	D7	D6	D5	D4	D3	D2	D1	D0
Primera Lectura												Segunda Lectura											

Control de la captura

Antes de realizar un pedido de lectura se debe establecer el tamaño del frame a leer, ya que la cámara entrega esa cantidad de píxeles antes de enviar el fin de frame, no atendiendo otra solicitud hasta terminar de enviar el frame completo. Antes de solicitar una lectura se construye un parámetro mediante el comando SendVideoFrame, formado con el modo de lectura Byte o Nybble, la decimación, cada cuantas filas o columnas se quiere leer, y los bits por píxel (24 o 32).

Síntesis en VHDL

Con las descripciones anteriores se modela un VHDL sintetizable [Cha97], donde se tienen las siguientes entidades: 1) *WriteDataReadParam*: realiza el *handshaking*, y la lectura de parámetros. 2) *Detectar*: detecta la cámara. 3) *Reset_Cam*: re-inicializa la cámara. 4) *Lectura*: captura una imagen (en modo Nybble o Byte).

Además, se usan componentes para la: 1) inicialización, 2) sincronización de tareas, y 3) solicitud de un frame con valores adecuados para cada modo de operación.

4 Etapa de análisis de imágenes

Esta etapa es la encargada de implementar el algoritmo de análisis de la imagen que está capturando la cámara. En cuanto empiezan a ser capturados los píxeles, se disparan los componentes de análisis de imagen. Dos componentes se han desarrollado para resolver problemas específicos.

4.1 Rectángulo y las coordenadas de sus vértices

El reconocimiento de un rectángulo claro en una imagen oscura y la identificación de las coordenadas de los vértices del mismo es el primer ejemplo analizado. Cabe destacar que la principal restricción del proyecto es la minimización de la memoria necesaria, por lo tanto se analiza la imagen a medida que es capturada (píxel a píxel). El proceso se realiza siguiendo los pasos: 1) Se captura un píxel por la cámara. 2) Se aplica un filtro de umbral a cada píxel, para decidir si forma parte del fondo o del rectángulo. 3) Se envía el píxel a una FSM encargada de reconocer los rectángulos.

Se reconoce un rectángulo tomando dos características de este patrón: que es un patrón convexo y que tiene 4 vértices. Basándose en estas características se desarrolla una FSM que las reconozca. En una imagen real estas características no son tan tajantes pues por cuestiones de iluminación los vértices se redondean, por lo tanto los vértices se encuentran en los cruces de las proyecciones de las líneas de delimitan un rectángulo. Por otra parte la convexidad no se cumple cuando una luz refleja en alguna zona de un borde del rectángulo. Se modifica la FSM reconocedora para que tome en cuenta estas variantes y además se toma en cuenta que cuando el rectángulo se encuentra rotado o no, pues la forma de proyectar los vértices difiere en la práctica (Fig. 3).

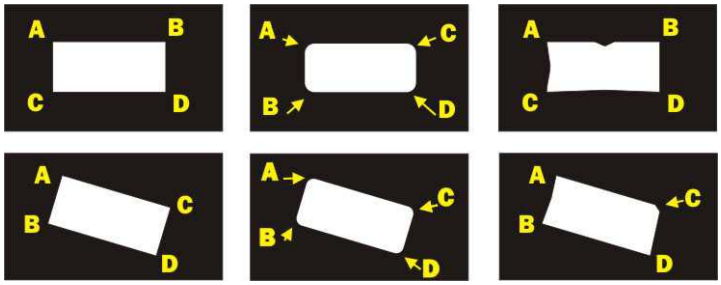

Fig. 3 - Casos generales de las coordenadas de un rectángulo

4.2 Reconocimiento de un Código de Barra y su decodificación

Aunque existen muchos tipos de Códigos de Barra, cada uno con su capacidad en cantidad de información que pueden codificar, los tipos más ampliamente difundidos en el mundo son los EAN-UCC [Ucc02] por sus características, potencialidad y simplicidad. Este patrón es muy rico en su conjunto de características y la información que brinda. A su vez EAN-13 [Ean02, Ean02a] es un patrón lineal o sea solo se necesita una línea para que este se encuentre en la imagen o no. Por lo tanto es un buen candidato a ser procesado por una FPGA con poca memoria, donde no es necesario

guarda la imagen sino ir procesando píxel a píxel y línea a línea. La Fig. 4 muestra las partes constituyentes del código de barra.
El símbolo de código de barra tiene la siguiente estructura de izquierda a derecha: 1) Una zona quieta izquierda. 2) Un patrón de guarda normal. 3) Seis caracteres símbolo de los conjuntos de número A y B. 4) Un patrón de guarda central. 5) Seis caracteres símbolo del conjunto de número C. 6) Un patrón de guarda normal. 7) Una zona quieta derecha. Donde, las zonas quietas son zonas claras (nada impreso), los patrones de guardas centrales y extremos son fijos y conocidos. Cada dígito pertenece un conjunto de códigos A, B y C, con paridad par o impar y donde C es el inverso del B. Además el dígito derecho es de Chequeo codificado como combinación de los anteriores.

Fig. 4 - Símbolo código de barra EAN-13

Se desarrollan dos maquinas de estados una reconocedora y otra decodificadora. A) La maquina reconocedora busca las características propias de un código de barra: zona quieta, guarda, cantidad fija de barras oscuras con tamaño máximo, y zona quieta. Si se llega a la zona quieta de la derecha se considera que se encontró un código de barra.
Cuando la maquina detecta una zona quieta y una guarda, la maquina decodificadora trata de decodificar el código utilizando un algoritmo recomendado por la norma ISO 15416 [Iso]. Cada vez que un píxel es capturado, el componente de lectura informa la posición en la imagen, que es usada para corrección de errores.

5 Referencias Bibliográfica

[Iso] ISO/IEC 15416. "*Information Technology – Automatic Identification and Data Capture Techniques – Bar Code Print Quality Test Specification – Linear Symbols*".

[Bro94] S. Brown, R. Francis, J. Rose, Z. Vranesic, "*Field-Programmable Gate Arrays*". Kluwer Academic Publishers,1994.

[Cha97] K. C. Chang, "*Digital design and modeling with VHDL and synthesis*". IEEE, 1997.

[EAN02] EAN International, Belguim, 2002, http://www.ean-int.org.

[EAN02a] "*General EAN-UCC Specifications V3.0*", 2002, http://www.ean-int.org/e_topnav2000.html.

[Esc01] Arturo de la Escalera Hueso, "*Visión por Computador, Fundamentos y Métodos*", Prentice Hall, Madrid, 2001

[Gon93] Rafael C. Gonzalez, Richard E. Woods, "*Digital Image Processing*". Addison Wesley, USA1993,

[Mey01] U. Meyer, "*Digital Signal Processing with Field Programmable Gate Arrays*", USA, 2001

[Pre70] J.M.S. Prewitt, "*Object Enhancement and Extraction*", Picture Processing & Psychopictorics, Likin, B. S. y Rosenfeld, A. Eds., Academic Press, New York, 1970.

[Rey00] Patrick Reynolds, "*The CQcam v0.90, free Color QuickCam (PC/Parallel) control program for various PC Unix*", 2000, http://www.cs.duke.edu/~reynolds/cqcam/.

[Rob65] L. G. Roberts, "*Machine Perception of Three-Dimensional Solids*", Optical & Electro-Optical Information Processing, Tippet, J.T., ed., MIT Pres, Cambridge, Mass, 1965.

[UCC02] Uniform Code Council, Inc., USA, 2002, http://www.uc-council.org