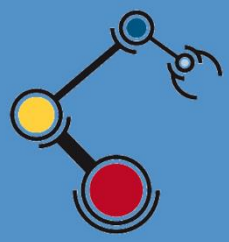
Escuela Politécnica Superior

Tracking de puntos para un sistema de control visual embebido en FPGA



Máster Universitario en Automática y Robótica

Trabajo Fin de Máster

Autor:
Alberto Torres Murcia

Tutor:
Gabriel J. García Gómez

Julio 2014

Universitat d'Alacant
Universidad de Alicante

RESUMEN

Autor: Alberto Torres Murcia

Título: Tracking de puntos para un sistema de control visual embebido en FPGA

Tutor: Gabriel J. García Gómez

Los sistemas de control visual obtienen la información de control a partir de una cámara. La imagen proporciona la referencia de control que permite guiar a un robot manipulador desde cualquier posición hasta una posición deseada en su entorno. El objetivo de este proyecto consiste en la detección de un patrón de cuatro puntos característicos que posibilite el tracking o seguimiento mientras permanezcan en el campo de visión de la cámara para facilitar el control visual aplicado sobre un robot. Para la captura de imágenes se ha empleado una cámara que se conecta a una Field Programmable Gate Array (FPGA) (1). La ventaja que se consigue al utilizar la FPGA frente a un procesador de propósito general secuencial es realizar un procesamiento en paralelo, lo que permite reducir los tiempos de respuesta con la información visual en el instante de capturar la imagen y realizar las operaciones necesarias en el control visual. La implementación software se ha desarrollado mediante el lenguaje de descripción VHDL y la herramienta ISE Design 14.7, permitiendo la portabilidad a cualquier FPGA. Se ha tenido en cuenta optimizar el consumo de recursos trabajando a la máxima frecuencia de reloj posible. El hardware empleado para el proyecto es la placa KC705 Eval Board que contiene una FPGA Kintex7 de Xilinx [7] y la cámara MV-D752-CL CMOS CameraLink de Photonfocus. El resultado de este tracking permitirá desarrollar aplicaciones de control visual embebido en FPGA.

Palabras Clave: FPGA, Xilinx, Procesamiento de imágenes, sistemas embebidos

Índice

Índice de figuras

1. INTRODUCCIÓN

1.1 Aspectos generales

Una de las principales técnicas utilizadas en la literatura para guiar un robot por visión es el control visual [3]. En estos sistemas de control la adquisición y el procesamiento de las imágenes en tiempo real es una cuestión crítica. Por tanto se requiere tiempos cortos para el procesamiento y altas frecuencias para la adquisición de imágenes, lo que permitiría al sistema aplicar las acciones necesarias de forma rápida a partir de las características extraídas.

Generalmente para implementar los algoritmos de visión se emplean plataformas basadas en algoritmos secuenciales empleando procesadores de propósito general, éstos no son los más óptimos desde el punto de vista de rendimiento ya que no ofrecen la suficiente potencia de cálculo debido a que el nivel de paralelización de los algoritmos de visión suele ser muy alto. Para conseguir cumplir con las restricciones de tiempo real y evitar un alto coste hardware y software especializado se requiere una alta frecuencia de procesamiento de píxeles, computación paralela y un hardware especializado. Por tanto las FPGA son idóneas para cumplir dichos requisitos debido a su arquitectura abierta y paralelismo de computación.

Los sistemas de visión, como se observa en la tabla 1, se dividen en tres niveles: nivel bajo, medio y alto. En el nivel bajo se realizan operaciones a nivel de píxel como filtrado o detección de bordes. Se realizan operaciones sencillas como sumas y productos con un alto número de píxeles. En el nivel medio se operan bloques de píxeles realizando operaciones de segmentación o etiquetado de regiones. Y en el nivel alto se realizan operaciones de alta complejidad como los algoritmos de matching.

NIVEL	CARACTERÍSTICAS	PROCESOS
BAJO	-Gran número de píxeles -Acceso a píxeles -Operaciones simples	- Captura/adquisición -Preprocesamiento
MEDIO	-Operaciones complejas -Acceso a bloques de píxeles	-Segmentación -Descripción -Reconocimiento
ALTO	-Operaciones de alta complejidad -Acceso aleatorio	-Interpretación

Tabla 1-Niveles de procesamiento en imagen

- **Captura o adquisición:** se obtiene una imagen digital mediante un dispositivo de captura.
- **Preprocesamiento:** se aplican técnicas como la reducción del ruido, realce del contraste, es decir, operaciones morfológicas simples.
- S**egmentación:** se divide en partes etiquetadas una imagen lo que permite analizarla.
- **Descripción:** se realiza una extracción de características para diferenciar las partes segmentadas.
- **Reconocimiento:** se realiza una identificación de cada parte segmentada asignándole una etiqueta.
- **Interpretación:** se asocia a un conjunto de partes segmentadas cierto significado para realizar agrupaciones.

Por tanto, como para procesar digitalmente las imágenes se requieren muchas operaciones a nivel de bit y que se ejecuten en el menor tiempo posible, conseguiremos mayor rendimiento si empleamos una plataforma hardware de propósito específico que con una arquitectura secuencial de un computador convencional. Con operaciones simples como filtros sobre imagen, el rendimiento es muy alto, para operaciones más complejas es necesario adaptar el algoritmo a la plataforma específica.

Una característica en control visual es una estructura, compuesta de píxeles, que se encuentra en un contexto lógico y que puede ser medida y extraída de la imagen. A menudo se corresponde con una característica física del objeto. La selección de buenas características es crucial para el funcionamiento de los sistemas de control visual.

La naturaleza de los parámetros puede ser geométrica o fotométrica. Algunos tipos de parámetros de características utilizados en control visual son [4]:

- **Principales características geométricas**:

  Las características extraídas en la imagen son puntos, líneas, segmentos, círculos, elipses, esferas, etc. El sistema de visión artificial debe extraer los parámetros adecuados de la primitiva geométrica y posteriormente aplicar el Jacobiano de la imagen adecuado. Las coordenadas de los puntos se pueden determinar a partir del centroide de los objetos presentes en la imagen. El centroide o centro de gravedad de un objeto es una característica muy utilizada y se define como el punto más representativo del objeto en la imagen, lo que resulta una aproximación de la localización del objeto en la imagen. Normalmente se representa la posición de la característica en un espacio cartesiano.
- **Patrones de intensidad:** permite detectar la localización de un determinado patrón de intensidad a lo largo de una secuencia de imágenes.
- **Snakes**: se tratan de contornos activos que se mueven y deforman de acuerdo a varias "fuerzas", y que tienden a adaptarse al objeto seguido, empleando funciones con forma del snake.
- **Contornos**: emplea información relativa al contorno de un determinado objeto presente en la imagen.
- **Momentos en la imagen**: utiliza momentos en la imagen.
- **Combinación de distintas características**: se extraen más de una característica, lo que resulta en una acción de control más robusta.
- **Información del modelo 3-D del objeto**: utiliza información del modelo del objeto a seguir lo que permite un sistema de control visual más robusto.
- **Esferas**: utiliza un modelo de proyección esférica para realizar el control visual.

1.2 Aspectos generales de la FPGA

En la implementación de un sistema electrónico disponemos de alternativas que nos permiten obtener un rendimiento alto sin mayores costes y manteniendo los requisitos del diseño. Los sistemas electrónicos se pueden aplicar perfectamente a un sistema en el que se procesan imágenes, en el que se requieren algoritmos con mayor coste computacional y potencia de cálculo. Existen diversos dispositivos para estos propósitos como son los dispositivos DSP (Digital Signal Processor), ASIC (Application-Specific

Integrated Circuits) y FPGA (Field Programmable Gate Array). Dependiendo de factores como el coste, potencia consumida o tiempo de diseño, será necesario decidir un dispositivo u otro.

Las FPGA han evolucionado tecnológicamente mejorando sus prestaciones técnicas, disponen de mayor número de recursos internos, lo que favorece el procesamiento de imágenes sobre estas plataformas hardware, convirtiéndolas en dispositivos adecuados para aplicaciones de sistemas en tiempo real. La evolución se debe a que los procesos de fabricación de los circuitos integrados han mejorado con el tiempo, así como el software de síntesis e implementación de las FPGA.

La evolución tecnológica de las FPGA ha permitido realizar proyectos de una manera más económica integrando en la FPGA los sistemas digitales y reduciendo el tiempo del proyecto.

Las FPGAs también tienen ciertas desventajas como el error producido por la codificación que se realiza habitualmente en coma fija, pocas aplicaciones se han desarrollado en coma flotante. Otra desventaja es que la codificación se realiza a bajo nivel mediante lenguajes de descripción hardware como VHDL o Verilog.

2. FPGA

Una FPGA (Field Programmable Gate Array) es un dispositivo semiconductor formado por bloques lógicos interconectados que pueden ser programados mediante un lenguaje de descripción hardware. Tienen utilidad en sistemas de fabricación automatizada, industria aeroespacial e incluso en la industria militar.

Las FPGA tienen la ventaja frente a circuitos comerciales preprogramados en que no tienen especificados los tiempos que tarda cada instrucción, es decir, es posible realizar un diseño estableciendo la frecuencia de reloj adaptada al circuito.

2.1 Lógica programable

En un hardware programable como es la FPGA, es posible disponer de un circuito genérico sobre el que programar una aplicación en particular. Las ALU frente a las FPGA tienen la limitación en que sólo puede ejecutar una operación a la vez frente a la posibilidad de implementar de manera paralela. Cualquier función lógica se representa en dos niveles: OR y AND. Las FPGAs están basadas en la arquitectura LUT, lo que significa que los bloques lógicos internos son más pequeños y en la arquitectura predominan las interconexiones de bloques. La programación de las FPGA

se lleva a cabo en celdas de memoria estática, con lo que la programación es volátil y se tiene que volver a cargar el programa cada vez que se enciende el dispositivo.

2.2 FPGA y procesamiento de imágenes

En una FPGA la lógica requerida por una aplicación se implementa mediante hardware separado para cada función, por ello se considera de procesamiento paralelo. Tiene la ventaja de obtener una velocidad propia de un diseño hardware siendo posible su reprogramación. Por tanto, son muy adecuadas para el procesamiento de imágenes, en concreto en niveles bajo e intermedios para poder explotar mejor el paralelismo.

En una arquitectura segmentada, cada bloque hardware se construye para cada operación de procesamiento de imagen. En un sistema síncrono, cada dato se pasa de la salida de un bloque de una operación a la entrada del siguiente. Si no es síncrono se emplean buffers intermedios entre las operaciones de datos.

El paralelismo espacial se explota construyendo múltiples copias de las operaciones implementadas y asignar diferentes particiones de la imagen.

El paralelismo lógico en una operación de procesamiento de imagen se adapta perfectamente a una FPGA, consiguiendo mejorar el rendimiento de los algoritmos de procesamiento de imagen de manera significativa gracias al hardware paralelo.

Ya que los datos de la imagen viajan en serie, si se implementan las operaciones de procesamiento en “stream”, el algoritmo puede resultar muy eficiente aprovechando los recursos de la FPGA.

El paralelismo en las operaciones permite bajar la velocidad del reloj significativamente. El cauce segmentado en una FPGA puede operar perfectamente a la misma frecuencia con la que se van sirviendo los píxeles, y por tanto, a menor frecuencia, menor potencia requerida por el sistema.

2.3 Dentro de una FPGA

En la FPGA, la lógica programable está diseñada como un conjunto de bloques de granularidad fina basados en una arquitectura LUT (LookUp Tables) como se puede observar en la figura 1. Los bloques lógicos se organizan en forma de malla interconectados mediante una matriz de enrutamiento programable permitiendo una configuración arbitraria. El núcleo de la FPGA y los dispositivos externos están conectados gracias a los bloques de E/S.

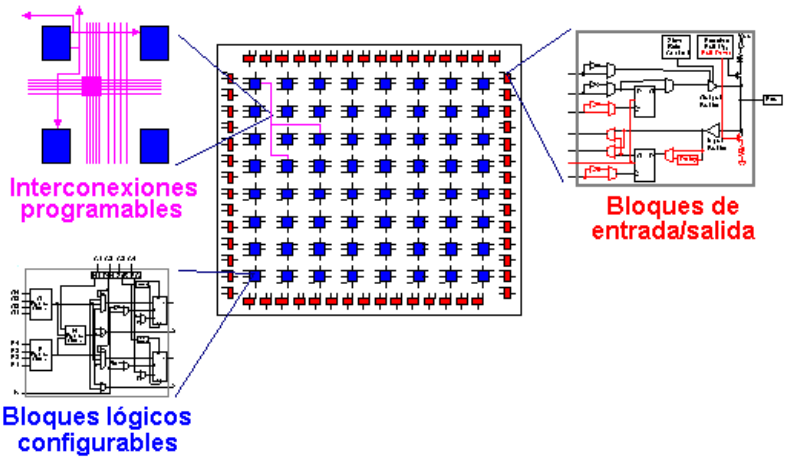


Figura 1 - Arquitectura interna de una FPGA

Además disponen de una distribución de señales de reloj en todas las partes permitiendo controlar la temporización de una señal de reloj con relación a una fuente externa y limitando la desviación de reloj.

2.4 Lógica

En una FPGA, la unidad más pequeña es una celda lógica, la granularidad más pequeña que es capaz de producir es un bit de salida. Generalmente está compuesta por una LUT (LookUp Table), con 3 entradas y un biestable en la salida.

Una LUT puede implementar cualquier función arbitraria con sus entradas y emular cualquier puerta lógica. Todas las celdas dentro de un bloque lógico comparten señales de control.

Las FPGA, en lugar de emplear memorias externas al chip, poseen pequeños bloques de memoria agregados al chip para almacenamientos intermedios y buffers. Se implementan de dos posibles maneras: utilizando bloques de memoria dedicados o empleando la estructura de las celdas lógicas para permitir usarlas como memorias de acceso aleatorio en vez de LUTs. La mayoría de bloques de memoria poseen 2 puertos, lo que simplifica la implementación de buffers FIFO y caches.

Disponen de una estructura heterogénea, mezclando bloques lógicos de grano fino y grueso para implementar las operaciones típicamente utilizadas.

2.5 Aplicaciones de las FPGA

Las aplicaciones actuales de las FPGAs son: el procesado digital de señal, sistemas aeroespaciales y de defensa, prototipado de ASICs, visión artificial, reconocimiento de voz, criptografía, bioinformática, emulación de hardware, redes neuronales, etc.

Las FPGAs competían con las CPLDs (Complex Programmable Logic Device) para reemplazar los circuitos digitales discretos, y con el tiempo han ido abarcando funciones cada vez más complejas, hasta incluso implementar SOCs (System-on-a-chip) e instalarse en sistemas reservados para chips DSP (Digital Signal Processing).

Se usan cada vez más en aplicaciones convencionales de computación de altas prestaciones HPC (High Performance Computing) como para calcular la transformada rápida de Fourier en FPGAs en vez de en microprocesadores. Aunque el uso de FPGAs en HPC está limitado por la complejidad del diseño con FPGAs, inconveniente que mejorará conforme haya disponibles mejores herramientas de programación de estos dispositivos.

2.6 Fabricantes

Los dos productores más conocidos de FPGAs de propósito general existentes en el mercado actual son Xilinx (Series Virtex y Spartan) y Altera (Series Cyclone y Stratix). Además, podemos encontrar otros competidores como Lattice, Actel, QuickLogic, Atmel, Achronix Semiconductor y MathStar Inc.

3. Lenguaje de descripción de hardware

La definición de lenguaje de descripción de hardware o lenguaje máquina es un conjunto de instrucciones codificadas que una computadora puede interpretar y ejecutar directamente.

HDL es el acrónimo de Hardware Description Language (Lenguaje de Descripción de Hardware), cuyo objetivo es programar un circuito electrónico.

El diseño de un proyecto en VHDL consiste en:

- Establecer las funciones que implementará el circuito.
- Codificar el programar con un lenguaje HDL.
- Análisis de la sintaxis y simulación del programa.
- Programar la FPGA y comprobar el funcionamiento.

Los lenguajes HDL son independientes del hardware, lo que significa que un código se puede usar en otros diseños más complicados y con otros dispositivos compatibles.

3.1 VHDL

VHDL [6] es un lenguaje que se ha diseñado para poder describir sistemas electrónicos digitales. Se creó en el programa VHSIC (Very High Speed Integrated Circuits) por el Departamento de Defensa de los EE.UU. en los años 1980. Fue estandarizado por el IEEE en Estados Unidos. Cubre las necesidades durante el diseño, realizando una descripción funcional o de comportamiento del circuito. Describe la estructura del diseño y declara las entidades y sub-entidades especificando una jerarquía y las interconexiones. Además permite simular el diseño y sintetizarlo con las herramientas correspondientes. Tiene la ventaja de eliminar la fase de prototipación de componentes reduciendo costes de diseño y producción.

3.1.1 VHDL para descripción de comportamiento

Una descripción VHDL está compuesta por entidades y sub-entidades, algunas se encuentran en componentes manufacturados, aunque es conveniente realizar una descripción funcional del componente, la cual, sólo se refiere a su funcionamiento.

3.1.2 VHDL como descripción de estructura

Un sistema electrónico digital consiste en un módulo con puertos de entrada y salida. Los valores de los puertos de salida son una función de los valores de los puertos de entrada y del estado del sistema.

Un sistema digital se puede describir con los componentes que lo forman. Cada componente es la instancia de alguna entidad, sus puertos están interconectados por señales. Es una descripción estructural o de estructura.

3.2 VERILOG

Verilog es también un lenguaje de descripción hardware para modelar sistemas electrónicos. Soporta el diseño, las pruebas y la implementación de circuitos analógicos, digitales y de señal mixta.

Fue inventado por Phil Moorby en 1985 en Automated Integrated Design Systems, posteriormente llamado Gateway Design Automation [5]. Con el tiempo se transformó en el lenguaje de descripción del mercado más comercial y de mayor relevancia. Además pasó a ser un lenguaje abierto y estandarizado por el IEEE. Las últimas versiones del lenguaje incluyen soporte para modelado analógico y de señal mixta.

3.2.1 Tipo de programa y estructura

El lenguaje es similar al lenguaje de programación C, dispone de un preprocesador como C y la mayoría de las palabras reservadas son similares. A diferencia de C, Verilog utiliza “Begin” / “End” en vez de usar llaves para delimitar un bloque de código.

3.3 JHDL

JHDL se basa en Java, emplea dos clases básicas, Logic y Wire. Los diseños son estructurales (bibliotecas) o de comportamiento (sólo simulación). El entorno de desarrollo permite generar netlists en HDLs a partir de los diseños JHDL estructurales. El compilador genera la estructura hardware que implementa la funcionalidad de un programa Java, por tanto, Java es la base para la generación de hardware.

A pesar el potencial de este lenguaje, ha tenido poco éxito en la comunidad de diseño hardware.

3.4 SystemC

Es un lenguaje basado en C++, se trata de un lenguaje de descripción de sistemas, el cual, permite un diseño de alto nivel, apropiado para particionamiento de sistemas, evaluación y verificación en la asignación de bloques para la implementación hardware o software, y también para la arquitectura y medición de las interacciones entre bloques funcionales.

4. Procesamiento de imágenes sobre una FPGA

En la figura 2 se puede observar una arquitectura abierta de software para el control visual sobre FPGA, este proyecto se centra en la parte de visión, la cual, se compone de cinco etapas persiguiendo el fin de realizar el procesado de las imágenes recibidas del *frame grabber* para extraer la información visual deseada.

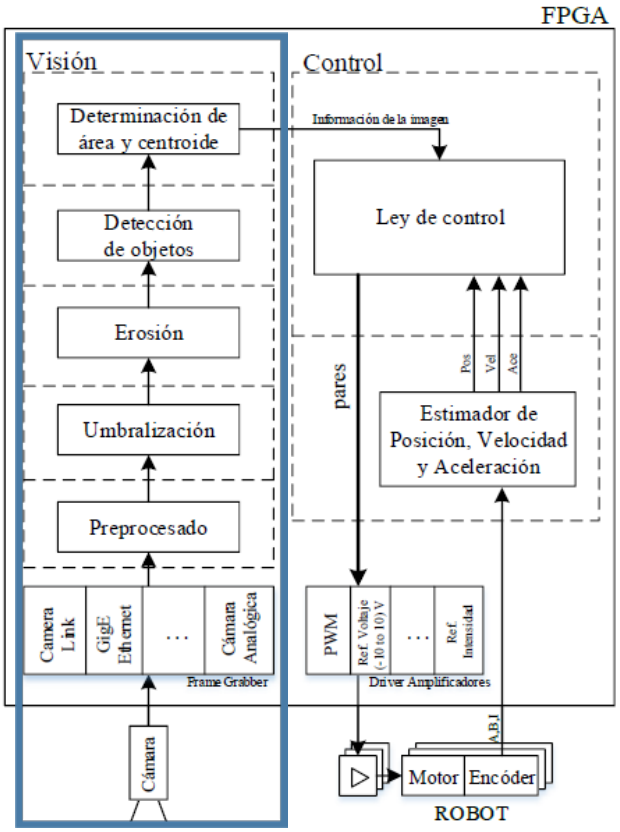


Figura 2 - Arquitectura software

El frame grabber es el módulo que depende del hardware empleado y se encarga de leer los píxeles de las imágenes de la cámara y por eso depende del protocolo que utiliza la cámara que se emplea en el sistema de control visual, en este caso, CameraLink.

Las etapas de visión están sincronizadas con el reloj de la cámara. La cámara utilizada envía los datos de la imagen a una frecuencia de 80 MHz, enviando dos pixeles a la vez en cada ciclo de reloj. Se ha configurado la cámara para capturar imágenes a máxima resolución (752x582 pixeles) y con velocidad de fotograma de 340 imágenes por segundo, por lo que el periodo de captura de una imagen entera es de 2.9 ms.

En la figura 3 se puede observar un cronograma que ilustra las señales que emite la cámara. FVAL (Frame Value) vale uno mientras se reciba un frame de imagen, LVAL (Line Value) vale uno mientras se reciba una línea de píxeles de la imagen, y DATA EVEN y DATA ODD son el par de píxeles que se envían cada ciclo de reloj.

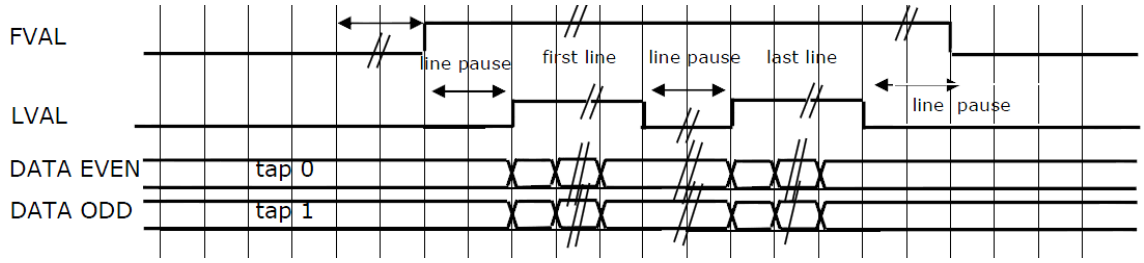


Figura 3 - Cronograma cámara MVD752

Para poder mostrar la imagen en un monitor VGA es necesario guardar el valor de los píxeles procedentes de la cámara en memoria, y mediante un controlador VGA, recuperar el valor de los píxeles para mostrarlos en pantalla. Se necesita el buffer intermedio debido a que la frecuencia a la que se sirven los píxeles de la cámara es mayor que la frecuencia a la que es capaz de leer el monitor VGA.

Se implementan los algoritmos de visión en un cauce segmentado de modo que se procesan los píxeles de la imagen que se van recibiendo sin la necesidad de almacenar la imagen en un buffer, suprimiendo la necesidad de esperar hasta la recepción de la imagen completa para comenzar a analizarla. De forma que se produce una reducción de la latencia de todo el sistema, a la vez que se realiza una optimización considerable de los recursos de almacenamiento y se disminuye el tráfico de datos.

A continuación se describe la función que realiza cada etapa:

1. **Preprocesado**: Dicha etapa es la encargada de aplicar operaciones preliminares con el fin de obtener un formato único de imagen independiente de la cámara utilizada, como la conversión de RGB a escala de gris, redimensionado de la imagen, filtrado de ruido, etc.

2. **Umbralización**: Es la función más simple en el procesamiento de imagen. Se aplica a imágenes monocromas y el resultado es una imagen binaria que tiene toda la información esencial del patrón. Esta función viene definida como:

$$g(x,y) = \begin{cases} 255, & f(x,y) \geq U \\ 0, & f(x,y) < U \end{cases}$$, siendo U el valor del umbral

3. **Erosión**: Es una operación morfológica cuyo resultado elimina los píxeles situados en los bordes de los objetos. Dicha operación se caracteriza por su capacidad para

eliminar el ruido. Se trata de una etapa opcional que no es necesaria si se ha aplicado algún filtro para reducir el ruido en la primera etapa de preprocesado.

4. **Detección de objetos**: Es la etapa más importante en el diseño, ya que su implementación consume la mayor parte de los recursos hardware de la FPGA. El objetivo de dicho módulo es el de detectar los píxeles conectados que forman un objeto en la imagen y etiquetarlos. Con el fin de optimizar la implementación sobre FPGA se ha seleccionado el algoritmo "Two-Pass Algorithm" [1], el cual realiza el etiquetado de la imagen en dos fases. En la primera fase se procesan los píxeles recibidos del módulo de Umbralización y se les asigna etiquetas provisionales almacenando dicha información en la memoria RAM. De forma paralela, se establece una tabla LUT (Look-Up Table) que determina las etiquetas que pertenecen a un mismo objeto. Esta tabla (denominada tabla de equivalentes LUT) se usa en la segunda fase para interpretar los datos de la imagen etiquetada de forma provisional que ha sido almacenada en la memoria RAM y para unir todas las etiquetas de dicho objeto. En la Figura 4 se muestra el funcionamiento de cada una de las fases de dicho algoritmo. La idea básica de la primera fase es: si el píxel recibido no es cero (fondo) se examinan los píxeles vecinos de arriba "A", "B" y "C" (en la línea anterior) y el píxel a la izquierda "D" (el píxel anterior). Si todos son ceros se asigna una nueva etiqueta al píxel actual, si no se le asigna la etiqueta menor (distinta de cero) de los píxeles vecinos "A", "B", "C" y "D". Una vez procesada y almacenada en el buffer toda la imagen, se guarda una copia de la tabla de equivalentes. De forma que en la fase 1 se puede procesar una nueva imagen mientras se procesa la imagen anterior en la fase 2 (en el cauce segmentado). Por otro lado, la tarea de la fase 2 es: leer los píxeles almacenados en la memoria RAM (Frame Buffer) e interpretarlos según la tabla de equivalentes LUT.

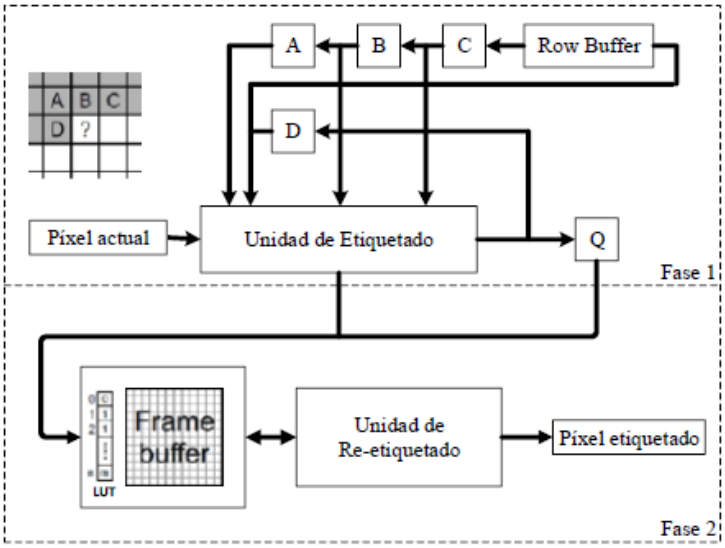


Figura 3 - Algoritmo de detección de objetos

En la Figura 5 se muestra un ejemplo práctico del algoritmo descrito para la detección de objetos.

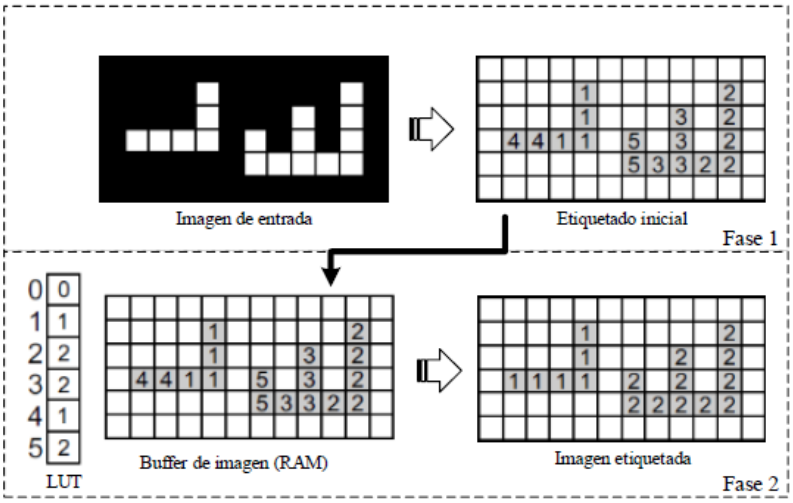


Figura 4 - Ejemplo del algoritmo para la detección de objetos

5. **Área y determinación del centroide**: El objetivo principal de toda la parte de visión de la arquitectura propuesta es determinar los centros de gravedad en el plano imagen de los objetos detectados y etiquetados en la etapa anterior. Existen dos aproximaciones para calcular las coordenadas del centro de un objeto en una imagen [2]:

- *Bounding Box*: donde se estima el centro de un objeto buscando las coordenadas XY de sus píxeles mínimas y máximas. Se define como:

$$x_c = \frac{\max\_Xpos + \min\_Xpos}{2}$$

$$y_c = \frac{\max\_Ypos + \min\_Ypos}{2}$$

- *Center-of-Mass*, donde se estiman los centros de gravedad teniendo en cuenta todos los pixeles que forman el objeto. Cabe destacar que la aproximación del centro de gravedad consume más recursos hardware para su implementación en la FPGA, sin embargo, se obtiene más resolución en el cálculo de los centros mientras que para la de Bounding Box se requieren menos recursos hardware pero los resultados obtenidos son muy sensibles al mínimo ruido que pueda sufrir la imagen.

$$x_c = \frac{\sum_i x_{pos\_i}}{Área\ del\ objeto}$$

$$y_c = \frac{\sum_i x_{pos\_i}}{Área\ del\ objeto}$$

donde el área de un objeto viene definida como el número de píxeles que lo forman, x_{pos_i}es la posición en el eje X del plano imagen de cada píxel del objeto e y_{pos_i} es la posición en el eje Y del plano imagen de cada píxel del objeto.

5. Arquitectura hardware

En la figura 6 se muestra la arquitectura hardware del sistema empleado.

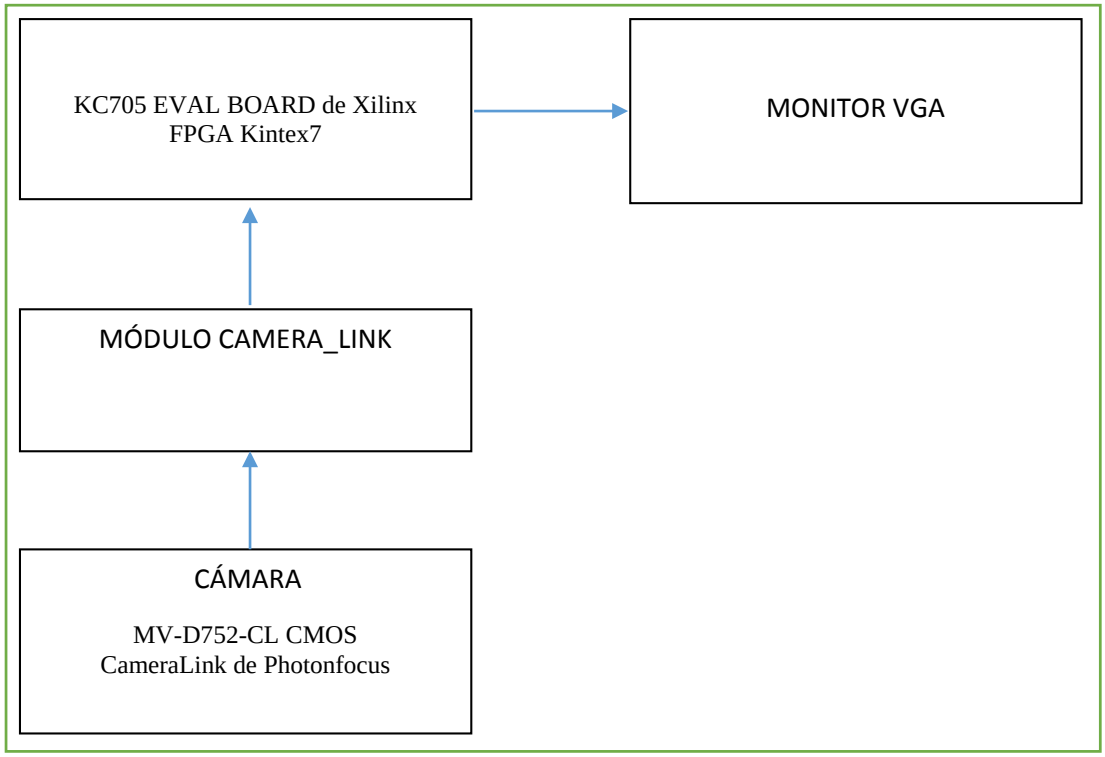


Figura 5 - Arquitectura Hardware

Se ha utilizado la cámara MV-D752-CL CMOS CameraLink de Photonfocus (figura 7) y la placa KC705 EVAL BOARD de Xilinx que contiene la FPGA Kintex 7 (figura 8).

Figura 6 -MV-D752-CL CMOS CameraLink

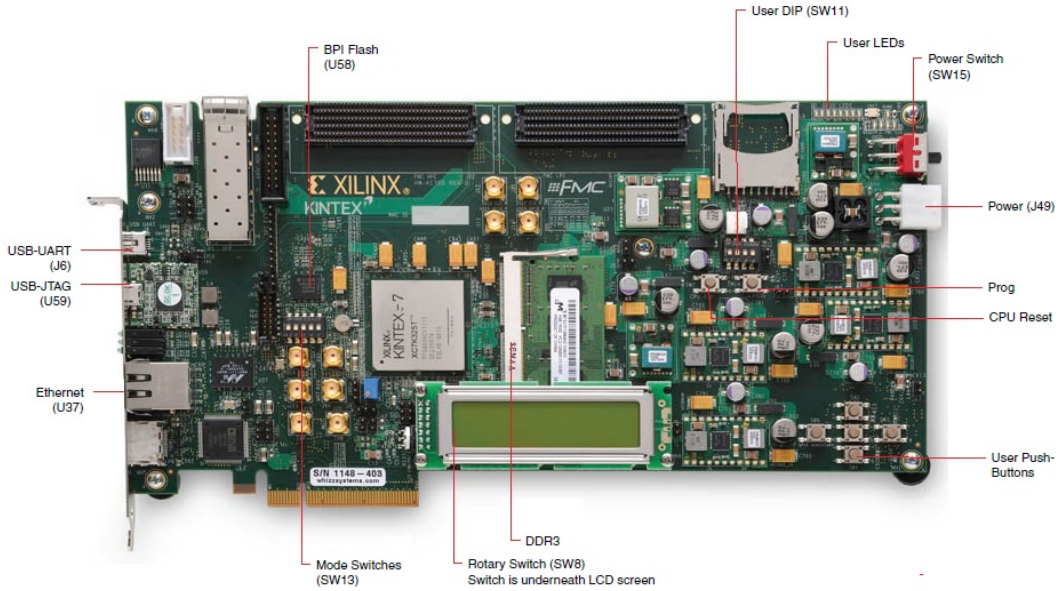


Figura 7 - KC705 EVAL BOARD de Xilinx

6. Pruebas

En la figura 9 se muestra la segmentación de los distintos algoritmos descritos en la sección 4. Como puede observarse, estos algoritmos se ejecutan en un cauce segmentado de forma paralela, una de las ventajas que ofrece la implementación de algoritmos sobre dispositivos FPGA. También se muestra la latencia requerida en cada etapa.

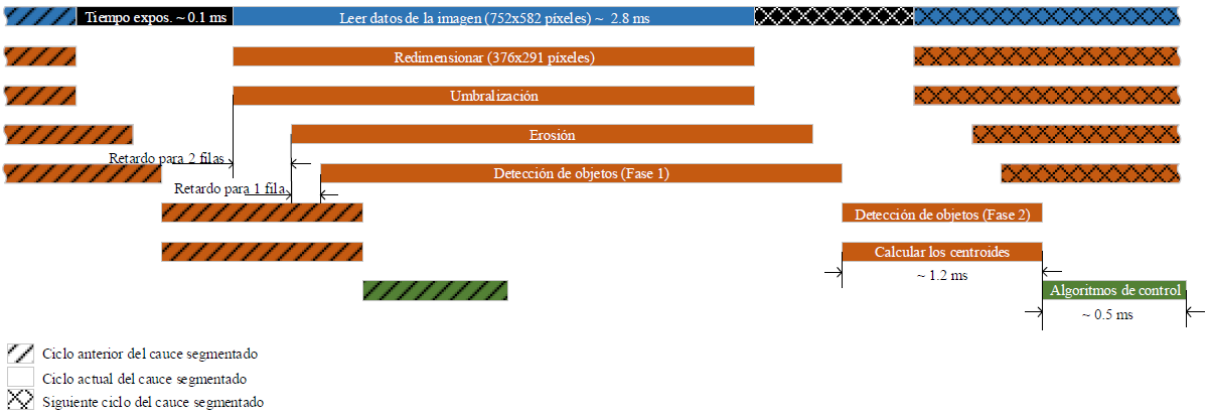


Figura 8 - Cauce segmentado del módulo de visión

Debido a que el tiempo de exposición de la cámara depende de las condiciones de iluminación y el material de la superficie de la escena, en los experimentos se ha hecho uso de un patrón de papel de color amarillo con fondo de tela de color negro. En la figura 8 se observa que el tiempo de exposición es de 0.1 ms y luego leer la imagen completa tarda 2.8 ms.

En la etapa de preprocesamiento se ha cambiado el tamaño de la imagen a la mitad por dos motivos: el primero es conseguir un píxel en cada ciclo de reloj y pasarlo al módulo siguiente, y el otro es reducir el ruido en la imagen. Se observa también que las etapas de preprocesamiento y umbralización se ejecutan en el mismo instante en que reciben los píxeles, mientras que la etapa de erosión tiene una latencia equivalente al tiempo de envío de dos líneas de la imagen. Esto es debido a la naturaleza de esta función morfológica, que necesita comprobar los píxeles vecinos del píxel actual. Lo mismo ocurre en la primera fase de la detección de objetos, donde la latencia es de una línea. Mientras, la segunda fase de esta etapa empieza una vez terminada la fase 1, y su latencia se produce por el tiempo de transferencia de datos entre la FPGA y la memoria. En los experimentos realizados esta latencia es de 1.2 ms utilizando la memoria interna (RAM Blocks) de la FPGA y el reloj de la placa 200MHz. La última etapa de visión calcula los centros de los objetos en paralelo con la fase 2 de la detección de objetos donde se ha empleado el algoritmo de centro de gravedad teniendo en cuenta que nuestra FPGA dispone de una cantidad bastante grande de recursos hardware. Una vez obtenidos los centros de los objetos se pasan a la parte de control para tomar las decisiones adecuadas y mover el robot de manera que siga el patrón.

En la Figura 9 se ha representado la imagen original, así como el resultado después de la fase de umbralización y de etiquetado.

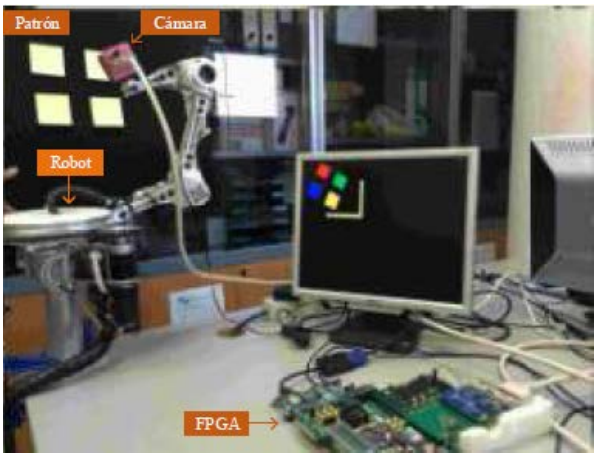


Figura 9 - Demostración de funcionamiento

7. Conclusiones

El sistema de visión, implementado en este proyecto, empleado en un sistema de control visual, se encarga de proporcionar la información necesaria para el cálculo de la acción de control. La arquitectura software implementada en la FPGA, es capaz de proporcionar la información visual a mayor velocidad que la que pueda proporcionar un sistema desarrollado sobre un procesador de propósito general como un ordenador. Para conseguirlo se ha implementado un cauce segmentado que permite obtener en paralelo el centroide de los objetos detectados en la imagen. Este módulo de visión implementado permitirá en una arquitectura abierta de control visual directo sobre FPGA enviar las coordenadas de las características en imagen a un módulo de control visual en el que se aplique la ley de control.

8. Agradecimientos

Quiero mostrar mi agradecimiento por la colaboración de los miembros del Grupo de Automática, Robótica y Visión Artificial, en especial a Gabriel, Jorge, Carlos y Aiman.

9. Lista de Referencias

[1] Bailey, D.G. (2011) “Design for Embedded Image Processing on FPGAs”, John Wiley & Sons: Singapore, Singapore, 2011.

[2] Bochem, A.; Kent, K.B.; Herpers, R., (2011) "FPGA based real-time object detection approach with validation of precision and performance", *22nd IEEE International Symposium on Rapid System Prototyping (RSP)*, vol. 9, nº 15, pp. 24-27.

[3] Chaumette, F. y S.A. Hutchinson, (2006) "Visual servo control. I. Basic approaches", *Robotics & Automation Magazine, IEEE*, vol. 13, nº 4, pp. 82-90.

[4] Garcia, G. J. (2010) “Control visual-fuerza autocalibrado para seguimiento de trayectorias en tareas cooperativas robóticas”, Tesis Doctoral, Universidad de Alicante.

[5] Verilog. http://www.verilog.com (visitado en Julio 2014).

[6] IEEE Standard VHDL Language Reference Manual," *IEEE Std 1076-2008 (Revision of IEEE Std 1076-2002)* , vol., no., pp.c1,626, Jan. 26 2009

[7] Xilinx KC705 Eval Board. http://www.xilinx.com/products/boards-and-kits/EK-K7-KC705-G.htm. (visitado en Julio 2014)