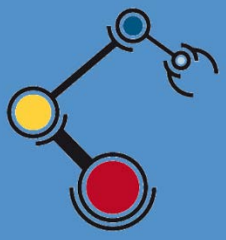
Escuela Politécnica Superior

Control visual embebido en dispositivo FPGA



Máster Universitario en Automática y Robótica

Trabajo Fin de Máster

Autor:
Beatriz Alacid Soto

Tutor:
Gabriel J. García Gómez

Universitat d'Alacant
Universidad de Alicante

Julio 2014

Resumen del proyecto

Autor: Beatriz Alacid Soto.

Título: Control Visual embebido en dispositivo FPGA.

Tutor: Gabriel J. García Gómez.

El objetivo principal del siguiente proyecto es el diseño e implementación de un controlador visual basado en imagen clásico, embebido en un dispositivo reconfigurable FPGA. Se propondrá una arquitectura abierta de control visual. Dentro de la arquitectura que se propone, se partirá de un sistema programado sobre la FPGA capaz de procesar una imagen capturada por la cámara. A partir de ésta imagen se obtiene la posición en imagen de los puntos característicos que se emplean para obtener el error del control visual. La principal aportación de este proyecto será la paralelización del algoritmo de control visual para poder programarlo en la FPGA, de forma que se puedan realizar tareas de control visual embebido.

Índice de contenidos

1 Introducción

Actualmente el control visual de robots es una de las tecnologías más empleadas con el fin de realizar el guiado de robots por visión. Dicho tipo de control emplea la visión artificial en la realimentación del bucle de control para controlar el comportamiento de un robot. Sin embargo, los sistemas de control visual, sobre todo en lo referente a la utilización de la visión artificial, requieren de alta capacidad computacional.

La adquisición y procesamiento de la información extraída de imágenes en tiempo real es una cuestión crítica en las aplicaciones de control visual. Por lo que, es esencial por un lado, una alta velocidad de captura de imágenes y, por otro lado, una baja latencia de procesamiento, puesto que el sistema de control visual debe tomar decisiones rápidas basadas en la información extraída de una escena. Para cumplir con las restricciones de tiempo real, el desarrollo de sistemas de control visual requiere generalmente un alto coste de hardware, así como software especializado. Esto representa un grave obstáculo para el diseño de sistemas de control visual en tiempo real. El procesamiento de imágenes en tiempo real impone serias exigencias relacionadas con una alta frecuencia de procesamiento de los píxeles, una potencia de computación masiva y paralela, y la utilización de hardware especializado (evitando implementar por software los algoritmos de visión artificial). Los procesadores de propósito general no siempre pueden proporcionar suficiente potencia de cálculo para cumplir con los requisitos de tiempo real debido a su naturaleza secuencial.

Por el contrario, las FPGA's (Field Programmable Gate Array) gracias a su arquitectura, paralelismo y flexibilidad de configuración, se están haciendo un hueco en el ámbito de la investigación como plataformas de ejecución para el procesamiento de imágenes en tiempo real. Este proyecto presenta una arquitectura embebida abierta de un controlador visual directo para robots industriales basada en FPGA. Esta arquitectura es extensible fácilmente para controlar cualquier robot manipulador multi-articular siempre que sus motores dispongan de encoders incrementales. De esta forma, el sistema será modular y flexible ante posibles cambios que puedan producirse como consecuencia de la incorporación o modificación de los drivers de control, o incluso a cambios de configuración en los sistemas de adquisición de datos y el control de los mismos. Actualmente es posible encontrar un gran número de controladores para robots que pueden clasificarse en propietarios, híbridos y abiertos. La mayoría de robots propietarios

presentan un controlador con una estructura cerrada en la que incluir hardware adicional o nuevas características de los controladores es difícil o imposible. Se habla de arquitecturas híbridas cuando la mayoría de las características, como son las leyes de control implementadas, son cerradas. Sin embargo es posible añadir fácilmente nuevos dispositivos hardware o sensores. Por último, en una arquitectura abierta el diseño de la arquitectura puede ser modificado por el usuario. Tanto el hardware como el software de la arquitectura son conocidos por el usuario y puede modificarlos sin dificultad. Por ello, la principal aportación de dicho proyecto es la propuesta de una arquitectura abierta para el control visual de robots. En este sentido, la reprogramabilidad y reconfigurabilidad que proporcionan las FPGA's permiten que el usuario pueda modificar cualquier elemento de la arquitectura (leyes de control, comunicaciones, sensores, etc.). Existen distintos proyectos para la definición de arquitecturas abiertas aplicadas a controladores. Este es el caso de OSACA (Open System Architecture for Control within Automation) creado para definir arquitecturas de control independientes del hardware [1]. En [2] se propone un sistema de control distribuido multi-agente de arquitectura abierta para una máquina de control numérico. Dentro del ámbito de las arquitecturas para el control de robots, cabe destacar los trabajos descritos en [3] en los que se muestran las características a tener en cuenta en el diseño de arquitecturas de control abiertas. Es de destacar trabajos como [4] en los que el uso de FPGAs es propuesto para el diseño de controladores de robots debido a su paralelismo y a la flexibilidad que aportan. Así, es posible encontrar arquitecturas para el control de robots que integran FPGAs en algún componente de su diseño [5]. En [6] se describe una arquitectura para el control de robots que es extensible fácilmente a cualquier robot y en la que parte de los controladores es implementado directamente en una FPGA. En comparación con estos desarrollos previos, en el presente proyecto se muestran los distintos componentes de una arquitectura abierta diseñada y optimizada para el control visual de robots.

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 Control Visual

2.1 Introducción acerca del control visual

El control visual se refiere al empleo de un sistema de visión con el fin de controlar el movimiento de un robot. Los datos necesarios para lograr dicho fin se adquieren desde una cámara que se fija directamente en un robot manipulador o móvil. Para ello, se utiliza la información captada de una escena por una o más cámaras conectadas al sistema de visión por computador, con el fin de controlar la localización del extremo del robot con relación a la pieza de trabajo.

Esta realimentación visual en tiempo real proporciona interesantes ventajas al sistema robótico. Una de ellas, quizás la más interesante, es la flexibilidad que se gana, permitiendo una mayor capacidad de interacción con el entorno, de modo que se dota al robot de la capacidad de reaccionar ante imprevistos. De este modo se consigue realizar

tareas en las que la orientación y las posiciones de las piezas de trabajo no están predefinidas.

Además, se obtienen movimientos de mayor precisión, puesto que el sistema de visión puede estar implantado en el extremo del robot, lugar en el que se realiza la acción. Por otra parte, se obtiene un incremento en velocidad de respuesta, ya que se tiene una imagen cada cierto intervalo corto de tiempo, permitiendo detectar problemas a gran velocidad.
Según dónde se encuentre situado el sistema de visión, los sistemas de control visual se clasifican como: **eye-in-hand** (cuando el sistema de adquisición de imágenes se encuentra el extremo final del robot), o como **eye-to-hand** (cuando el sistema de adquisición se encuentra montado en una referencia externa al robot de manera que se puedan visualizar los movimientos del robot, y así obtener la información necesaria para realimentar el sistema de control). En este proyecto la cámara se encuentra unida al extremo del robot, y los movimientos del robot se traducen en movimientos de la cámara.

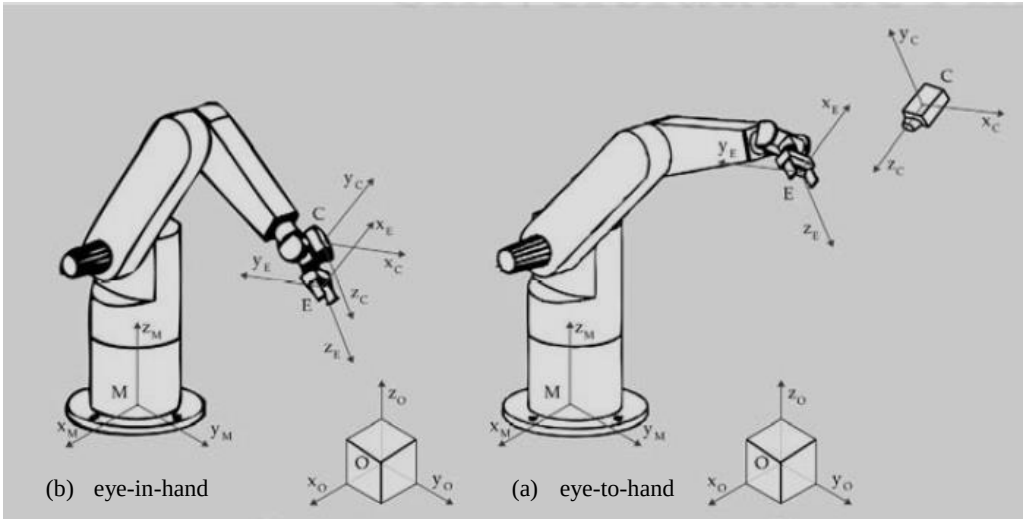


Figura 1: Esquemas de la configuración de la cámara: a) Configuración de cámara montada en el extremo y b) configuración de cámara externa

2.2 Control visual basado en posición

En control visual, se puede distinguir entre el basado en posición y el basado en imagen [7]. El primero trata de obtener las coordenadas del extremo del robot y las del objeto con el que se desea interactuar a través del sistema de visión. Para ello es necesario un modelo geométrico del entorno. A partir del error, dado por la diferencia entre la posición deseada y la posición obtenida (del objeto) en el procesamiento visual, se realimenta el bucle de control para llegar a la posición deseada. La principal desventaja de dicho tipo de control, es la ineficacia en entornos no estructurados.

Por otra parte, es necesario el conocimiento preciso del modelo cinemático y geométrico del robot, puesto que en caso de encontrar errores en dichos modelos, se obtendrían graves errores en el posicionamiento. Otra desventaja es la importancia del correcto funcionamiento de calibración de las cámaras presentes, ya que a partir de ellas se obtiene la posición del objeto respecto al sistema de referencia de la cámara.

A diferencia del anterior, el control visual basado en imagen sí permite trabajar en entornos no estructurados. Por lo que la sensibilidad a errores de calibración es menor. Por otra parte, ya no son necesarios los cálculos de las coordenadas cartesianas del objeto para llevar al robot hacia la posición deseada.

En el control visual basado en imagen se trabaja con las características de la imagen obtenidas por el sistema de visión. De esta forma, se tiene una serie de características en la imagen captada cuyos valores se desean modificar. Para ello, se parte de una imagen deseada en la que se ha obtenido el valor de las características, y que no es más que la imagen capturada por el sistema de visión en la posición objetivo.

En este caso, la función de error que realimenta el bucle de control es la diferencia entre los valores de las características deseadas y las iniciales (u obtenidas en cada momento). Puesto que ya no es necesario el cálculo de la localización del objeto, se reducen los errores de calibración y al mismo tiempo, se reducen los tiempos de procesamiento.

En este caso, para el proyecto que se describe en esta memoria, se ha utilizado para la primera parte, el control visual basado en imagen con una configuración eye-in-hand, el cual se refiere a la localización del sistema de adquisición de imágenes en el efector final del robot.

De modo que los movimientos que realice el robot se verán reflejados en dicho sistema de adquisición, para así tomar las decisiones oportunas para el guiado de dicho robot según la evolución de las características observadas en la imagen.
Para las características se utilizará un conjunto de cuatro puntos, de los que se conocerá la distancia hasta la cámara en la posición objetivo.

2.3 Control visual basado en imagen

En cuanto a los sistemas basados en imagen, el control visual se realiza directamente a partir de las características extraídas por el sistema de visión, que representan la proyección del objetivo a alcanzar en el plano imagen. De forma que la entrada al regulador será una comparación entre las características observadas y las deseadas o de referencia.

El control visual basado en imagen nos permite realizar el guiado de robots donde la ley de control se expresa en términos de características en las imágenes. Es decir, se minimiza el error entre las características de la imagen medidas y las deseadas. De esta forma, los sistemas de control visual basados en imagen implican la determinación de una función de error, **e**, la cual valdrá cero cuando la tarea se haya desarrollado correctamente y haya llegado a su fin. En dicho momento, las características observadas en la imagen, **s**, se corresponderán con las características deseadas $\mathbf{s}_d$. De modo que la acción de control se calcula en el espacio 2-D de la imagen.

En una aplicación típica de control visual en la que el robot debe posicionarse a cierta distancia respecto a un objeto del espacio de trabajo. De forma que, el robot describirá una determinada trayectoria a lo largo de la cual las imágenes captadas por el sistema de visión se irán modificando progresivamente. Es el controlador el que se encargará de ejecutar las acciones oportunas sobre el robot, de manera que las características observadas en las imágenes se vayan aproximando progresivamente a las deseadas, es decir, a la entrada de referencia del bucle de control.

Generalmente, las características de imagen que se utilizan en control visual son formas geométricas elementales (como pueden ser puntos característicos o esquinas) que permiten reconocer en una imagen la proyección de un objeto. Un ejemplo de la típica

imagen captada por un sistema de control visual basado en imagen es la que se puede observar en la figura 2.

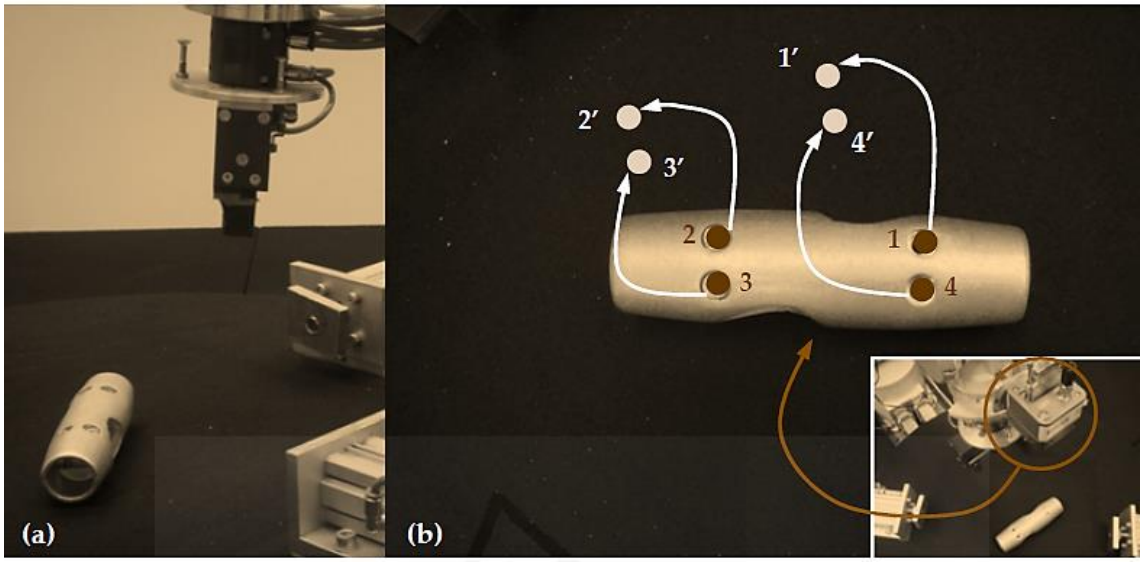


Figura 2: Imagen de las características captada por el sistema de control visual: a) Imagen captada con cámara externa al robot. b) Imagen captada con cámara en el extremo del robot.

La Figura 2.a) es captada por un sistema de cámara externa fija. Se puede observar la presencia de la pinza del robot, la cual se acerca al agarre de una pieza. La extracción de características sobre dicha figura permite la realimentación del sistema. De forma que la diferencia entre estas características y las deseadas se corresponde con la función de error que el controlador debe ir anulando progresivamente.

En cuanto a la Figura 2.b) se muestra una imagen obtenida con una cámara colocada en el extremo del robot. En dicha imagen se han mostrado las posiciones de las características actuales (1,2,3 y 4) y las deseadas (1’,2’,3’ y 4’). La diferencia entre la posición de ambas características es el error, e, que el sistema debe ir reduciendo progresivamente. Por lo que, las acciones de control del regulador harán que el robot se mueva con el objetivo de que las características observadas en la imagen se vayan acercando progresivamente a las deseadas. De manera que una vez que la función de error se anule, es decir, coincidan las características actuales con las deseadas; se supondrá que la tarea ha finalizado.

Generalmente, los objetos son a menudo simplificados al máximo para la realización del control visual basado en imagen. Para la extracción de características se suelen utilizar

técnicas como el marcado de puntos característicos en el objeto sobre un fondo uniforme (ver Figura 3). El tratamiento de la imagen se limita a la realización de una detección de dichas características y el cálculo del centro de gravedad de éstos con el fin de determinar el centroide de las características.

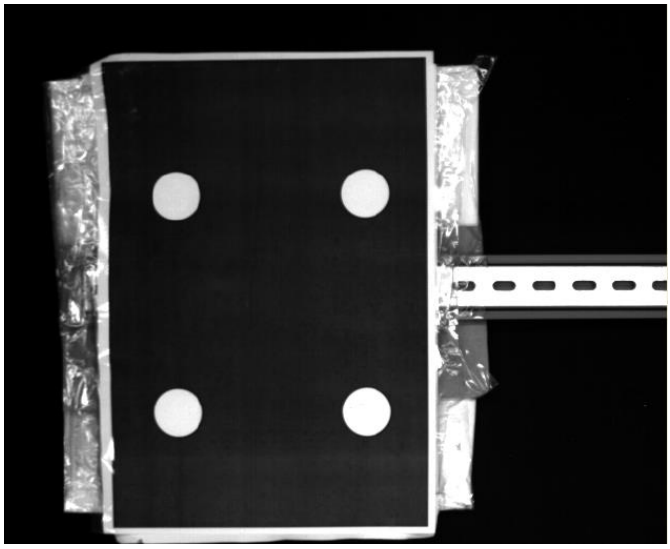


Figura 3: Ejemplo de un posible patrón para el control visual basado en imagen.

2.4 Control visual indirecto

Los sistemas de control visual clásicos parten de la información visual obtenida por la cámara situada en el extremo del robot (configuración eye-in-hand) para calcular en cada iteración una nueva posición del robot que acabe guiándolo hacia la posición final deseada. Para ello, el controlador obtiene entre iteración e iteración una nueva velocidad de la cámara que minimiza el error calculado como la resta entre la posición de las características visuales actuales medidas en el plano imagen y la posición de las características visuales deseadas. Al resultar conocida la posición relativa entre la cámara y el extremo del robot, es posible transformar las velocidades de la cámara proporcionadas por el controlador a velocidades del extremo del robot. Ésta es la base del control visual basado en imagen con configuración eye-in-hand. Dado que la acción de control del sistema es una velocidad del extremo, es el controlador interno del robot el que debe calcular qué pares articulares se deben aplicar para que el extremo del robot se desplace realmente a esa velocidad calculada.

2.5 Control visual directo

En el control visual directo se calcula y envía los pares y fuerzas articulares que deben guiar al robot hacia la posición requerida. El resultado es un control mucho más rápido y preciso que si se utiliza un control visual indirecto clásico. El principal inconveniente de estos controladores es que se debe trabajar directamente con los parámetros dinámicos no lineales del sistema robótico. Cuando se diseña un robot, se conocen de forma precisa sus parámetros dinámicos (masas, localización del centro de gravedad, términos de inercia y parámetros de fricción). Sin embargo, estos parámetros son desconocidos en la mayoría de robots industriales de carácter comercial. En ausencia de fricción u otras perturbaciones, la dinámica de un robot manipulador puede escribirse como [8]:

$$M(q)\ddot{q} + \mathrm{C}(\mathrm{q}, \dot{q}) + \mathrm{g}(\mathrm{q}) = \tau \qquad (1)$$

donde q ϵ R ^{nx1} es la configuración articular, $\dot{q}$ ϵ R ^{nx1} representa las velocidades articulares y $\ddot{q}$ ϵ R ^{nx1} las aceleraciones articulares, τ ϵ R ^{nx1} es el vector de fuerzas y pares articulares aplicados, M(q) ϵ R ^{nx1} es la matriz de inercia (simétrica y definida positiva), C(q, $\dot{q}$) ϵ R ^{nx1} es el vector de pares y fuerzas de Coriolis y g(q) ϵ R ^{nx1} es el vector de fuerzas y pares gravitacionales.

El control visual descrito en [9] es un control visual basado en imagen, dado que la tarea del robot se especifica en el plano imagen en términos de los valores de las características de la imagen correspondientes a las posiciones relativas entre el robot y el objeto. La aproximación que se sigue en [9] viene motivada por la filosofía de control de Jacobiana traspuesta introducido en [10]. La ley de control propuesta viene dada por:

$$\tau = \mathrm{J}(\mathrm{q}, \mathrm{s}, \mathrm{Z})^T K_P\, e - K_v \dot{q} + g(q) \qquad (2)$$

donde Kp ϵ R ^{2Mx2M} y Kv ϵ R ^{nxn} son las constantes proporcional y derivativa respectivamente. Estas constantes son matrices simétricas definidas positivas elegidas por el diseñador del controlador, M es el número de características del objeto en la imagen y *n* es el número de articulaciones del robot. Dentro del primer término de (2), **e** = (**s** − **s**d)

es el error en imagen, siendo **s**d el conjunto de características de la imagen en la posición deseada del robot respecto al objeto y **s** el conjunto de las características de la imagen en la iteración actual.

J(**q**,**s**,**Z**)^T se define como la traspuesta de la matriz Jacobiana, que en [REFERENCIA 6] se define como:

$$J(q,s,Z) = L_s(s,Z) \cdot J_g(q) \in R^{2Mxn} \quad (3)$$

donde L_s (s,Z) ϵ R ^{2Mx6} es la Jacobiana de la imagen para las características elegidas [11] y depende de la posición actual de las articulaciones del robot, ya que se emplea una configuración eye-in-hand [12]; y J_g (q) ϵ R ^{6xn} es el Jacobiano del robot en su extremo [13] (también se calcula a partir de la posición actual de las articulaciones):

$$\begin{bmatrix} V_c \\ \omega_c \end{bmatrix} = J_g(q) \cdot \dot{q} \quad (4)$$

El segundo término de (2) ($-K_v \dot{q}$) se corresponde con la acción derivativa y requiere que se disponga en cada iteración de las velocidades de las articulaciones del robot, $\dot{q}$. Mientras que el término **g**(**q**) es el término de pares gravitacionales del modelo dinámico del robot. Será, por tanto, necesario su cálculo para obtener los pares y fuerzas a aplicar al robot para minimizar el error en imagen. Este término de la dinámica del robot depende únicamente de la posición actual de las articulaciones, **q**.

Para el cálculo de la ley de control descrita en (2) se necesita una serie de datos en cada iteración. En el siguiente apartado se describe la arquitectura propuesta para proporcionar esta información al sistema en tiempo real.

3 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.

3.1 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.

3.2 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.

3.3 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 4. 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.

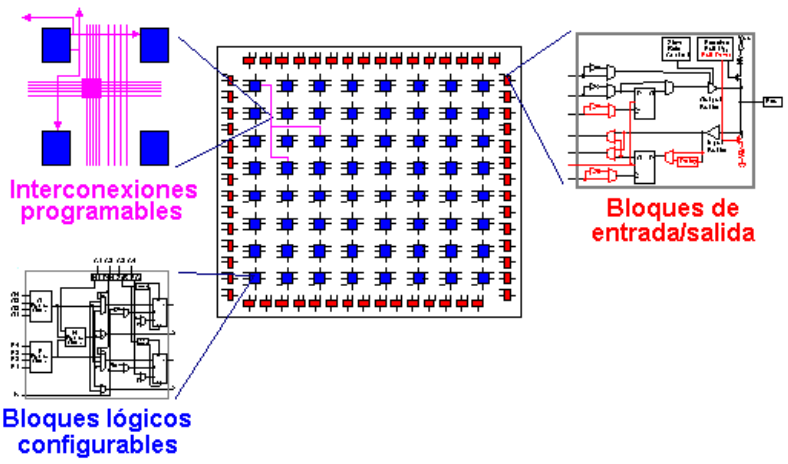


Figura 4 - Arquitectura interna de una FPGA

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.

3.4 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.

3.5 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.

3.6 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.

4 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.

4.1 VHDL

VHDL 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.

4.1.1 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.

4.1.2 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.

4.2 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 [14]. 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.

4.2.1 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.

4.3 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.

4.4 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.

5 Arquitectura abierta de control visual directo

La arquitectura propuesta a continuación permite realizar el guiado de un robot manipulador por medio de un control visual directo. El objetivo perseguido es la implementación sobre una FPGA de un esquema general que permita el control visual de cualquier tipo de brazo robótico utilizando una cámara cualquiera [15]. A continuación se detalla tanto la arquitectura hardware como la arquitectura software. En la primera se definen los componentes de la arquitectura generalista y los elementos particulares que se han utilizado en la validación de la arquitectura propuesta. Por otra parte, en la arquitectura software se describen los módulos dependientes de la parte hardware que se deben desarrollar para la utilización del sistema en el guiado de otros tipos de robots y con la utilización de cámaras de otro tipo.

5.1 Arquitectura hardware

En la figura 5 se observa el esquema de la arquitectura hardware compuesta por cinco partes principales: un robot manipulador, una cámara rápida situada en el extremo del robot, una terminal de interfaz (PC), amplificadores de potencia para el control de los motores del robot y la FPGA (en la que se implementan los algoritmos de visión y control

visual). Además, es necesario disponer de adaptadores de tensión entre los componentes citados anteriormente y la propia FPGA.

En este caso, para la implementación del diseño propuesto se ha seleccionado una FPGA Kintex7 de Xilinx (placa KC705 Eval Board)[16]. Dicha placa contiene dos conectores FMC, los cuales facilitan el acceso a las entradas/salidas de la FPGA, y un puerto serie UART para la comunicación entre la propia FPGA y el PC terminal.

La cámara seleccionada es la MV-D752. Se trata de una cámara CMOS monocromo, muy compacta en formato CameraLink con resolución 752 x 582, la cual está equipada con un sensor 2/3" que puede trabajar a 350 imágenes por segundo a su máxima resolución. Dicha cámara posee un rango dinámico de 120db el cual le permite resolver aplicaciones de visión sin depender tanto de la iluminación, ya que podemos realizar el análisis al mismo tiempo de zonas muy oscuras y zonas muy brillantes.

El robot que se ha empleado en esta implementación es COOPER [17], un brazo de 3 articulaciones de tipo RRR. Dicho robot está diseñado y construido íntegramente en la Universidad de Alicante por el grupo de investigación AUROVA. Está compuesto por tres motores Maxon acoplados en cada una de las articulaciones por medio de un sistema reductor planetario. Por otra parte, en la sensorización de la estructura robótica se integran encóders relativos al eje de los motores y un sensor inductivo en cada articulación del robot. Para realizar el control de los motores se utilizan unos servoamplificadores de Maxon (modelo ADS 50/5). Dichos servoamplificadores toman como referencia una tensión analógica de 10V a -10V, generada por un conversor digital analógico DAC situado entre la FPGA y los amplificadores.

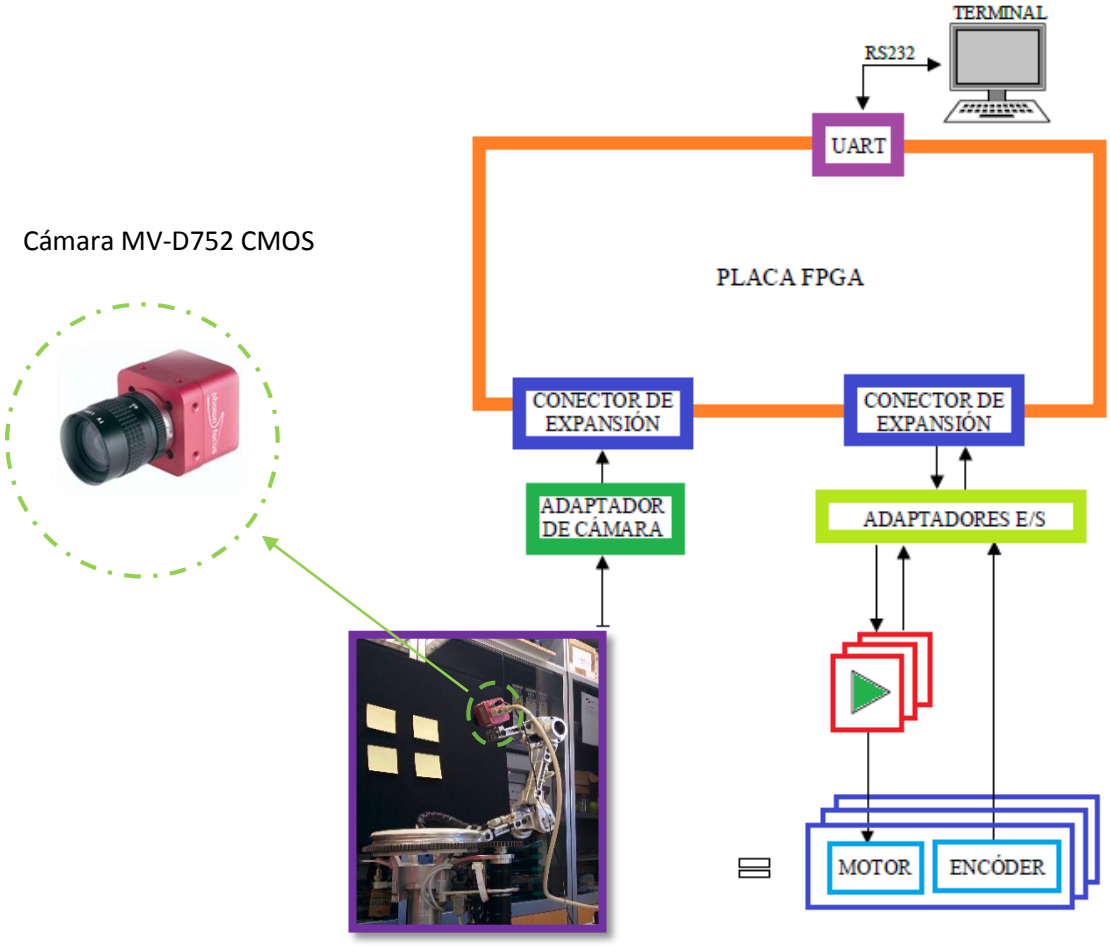


Figura 5: Arquitectura hardware

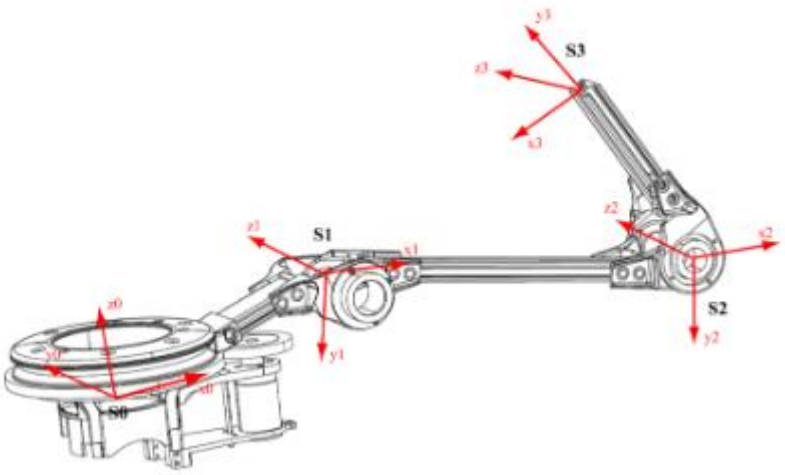


Figura 6: Sistema D-H del robot COOPER

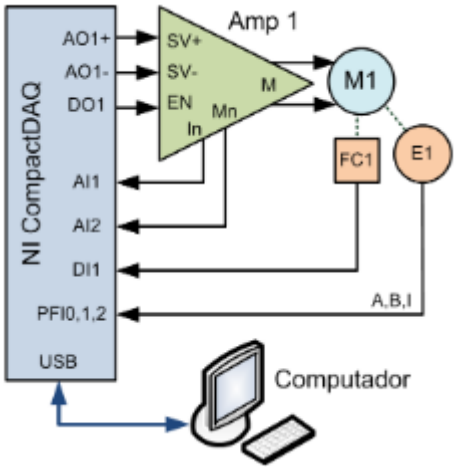


Figura 7: Esquema de control de los motores.

A partir de la arquitectura hardware descrita anteriormente se pretende evaluar el comportamiento de la arquitectura software abierta desarrollada sobre la FPGA.

5.2 Arquitectura Software

En un sistema de control visual es necesaria la existencia de un subsistema de visión muy sofisticado con el fin de lograr un controlador robusto dotado de la capacidad de realizar el procesamiento de imágenes en tiempo real a una velocidad de fotograma relativamente alta. De lo contrario, si dicho sistema no alcanza la suficiente potencia computacional ni una alta frecuencia de transferencia de datos, la ejecución del algoritmo se puede ver comprometida. En conclusión, la reacción de los robots con una frecuencia baja en el bucle de control sería lenta y por tanto, podrían fallar en la ejecución de tareas en los que el tiempo es un requisito estricto.

Pretendiendo la resolución de dicho problema de eficiencia y rendimiento del sistema de control visual se propone una arquitectura abierta de software para implementar sobre FPGA's. En la Figura 8 se muestra el esquema completo del software implementado. Con el fin de obtener una mayor flexibilidad del sistema, se han realizado divisiones de todo el proceso en subprocesos, implementando la funcionalidad de cada subproceso en un módulo independiente. Dichos módulos están sincronizados y se ejecutan de forma paralela aprovechando las ventajas de la arquitectura de las FPGA's.

Además, puesto que se pretende que la arquitectura sea genérica, se han aislado los módulos que dependen del hardware empleado de los módulos generales. A continuación se describen los diferentes módulos particulares que componen la arquitectura software:

a) ***Frame Grabber***: Dicho módulo es el encargado de la lectura de los datos de las imágenes que proporciona la cámara. Depende del protocolo que utilice la cámara que se emplea en el sistema de control visual, puede ser: CameraLink, GigE Ethernet, etc.

b) **Controladores de los amplificadores**: Este módulo se encarga de realizar la conversión de los pares resultantes de la etapa de control en señales adecuadas para controlar los amplificadores de potencia de los motores del robot. Por ello, dependen del tipo de señal de entrada que toman como referencia dichos amplificadores. En general, la mayoría de amplificadores de potencia para motores DC utilizan como entrada una señal PWM debido a su sencillez, pero también existen otros modelos de amplificadores que toman como referencia una señal analógica que puede ser voltaje o corriente.

El resto de módulos son genéricos e independientes del hardware utilizado. Se han clasificado en dos partes: visión y control. En primer lugar, se describe el módulo de visión, detallando las propuestas para conseguir un bloque funcional abierto capaz de proporcionar a los sistemas de control visual la información relevante en el plano imagen.

Como se puede observar en la Figura 8 la parte de visión se compone por cinco etapas persiguiendo el fin de realizar el procesado de las imágenes recibidas del *frame grabber* para extraer la información visual deseada. Aunque en este proyecto no se desarrolla el módulo de visión, se describen sus características de cara a detectar los principales requerimientos del sistema de control. La implementación de los algoritmos de visión se produce 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

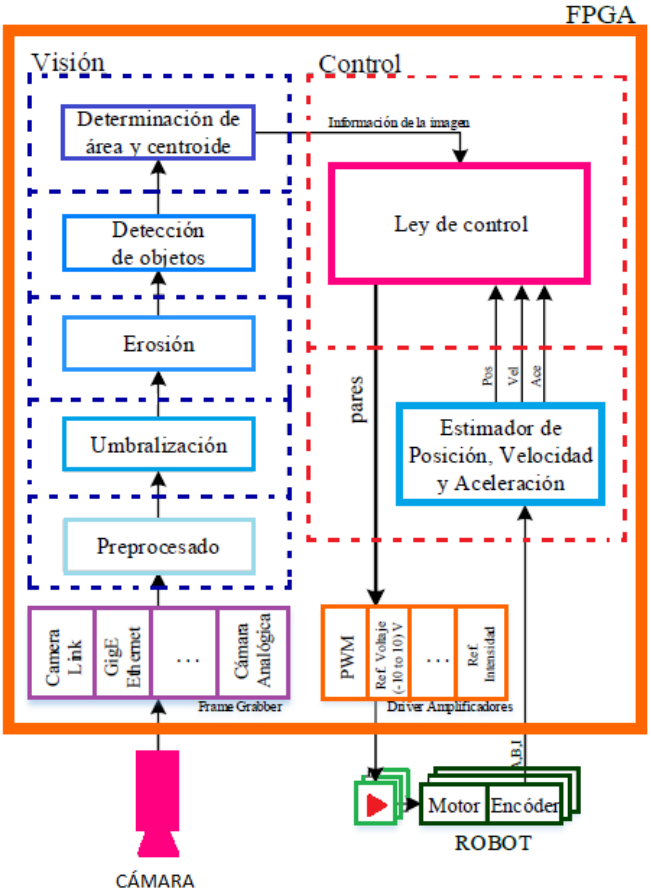


Figura 8: Esquema completo de la arquitectura software implementada.

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", el cual realiza el etiquetado de la imagen en dos fases. 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).

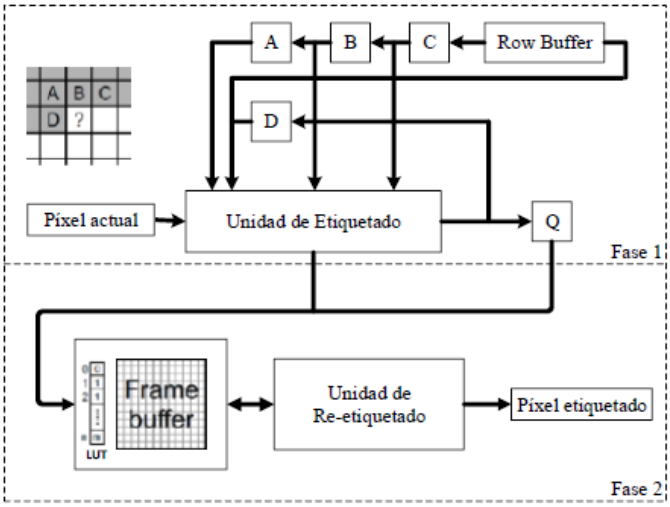


Figura 9: Algoritmo de detección de objetos.

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

Anteriormente se ha descrito la arquitectura software y se ha visto que, con el fin de obtener una arquitectura lo más genérica y abierta posible en la parte de control, se diferencia entre dos módulos: en uno se encapsula la "Ley de control" y en el otro módulo, se realizan los cálculos necesarios para aplicar la Ley de control seleccionada para el módulo anterior.

Para el módulo de la "Ley de control" se necesita conocer tanto la información de la imagen como la posición, la aceleración y la velocidad [18]. A partir de dichos parámetros y mediante la aplicación del control se obtienen los pares que se deben enviar al robot.

En base a los estudios y a la experimentación realizada es conocido que los principales problemas que se encuentran son el tiempo y los recursos invertidos en la realización de los cálculos necesarios para la aplicación de la "Ley de control". Operaciones como: el cálculo de la matriz inversa, producto de matrices de números decimales, etc. Una de las principales ventajas que se obtienen de la implementación sobre una FPGA es que permite la ejecución de diversas operaciones en paralelo, lo que se traduce en una optimización considerable en los tiempos de ejecución de las operaciones. Es decir, se mejora el tiempo final de ejecución. Por otra parte, algo a tener en cuenta es que dicha mejora en los tiempos de ejecución se traduce en un incremento en el número de recursos necesarios para ejecutar dichas operaciones.

Existen diversas implementaciones ya realizadas que se pueden utilizar, como el HDL Coder (un generador de código Verilog y VHDL para FPGA's y ASIC). Para las FPGA's de Altera (sólo algunas versiones) existen funcionalidades ya implementadas para operaciones con matrices de decimales o incluso la funcionalidad para la obtención de la matriz inversa.

El camino seguido en el presente proyecto es la búsqueda del equilibrio entre los tiempos y los recursos, es decir, optimizar tiempos y en la medida de lo posible, recursos; por lo que pese a que ya existen diversas implementaciones que se pueden adaptar al proyecto

sin la necesidad de implementarlas desde cero, se decidió implementarlas buscando tanto la rapidez como la optimización de recursos.

Por otra parte, en cuanto al módulo "Estimador de posición, velocidad y Aceleración" es necesario conocer información que proporciona el encóder de cada motor, a partir del cual se obtiene la posición, velocidad y aceleración necesarias.

El cálculo de la posición actual está basado en el número de revoluciones completas y el ángulo de rotación. Se utiliza un contador de revoluciones y un contador para el ángulo (Figura 10). La fórmula que determina la posición es:

$$Posición \frac{(rev \cdot m + ángulo) \cdot \theta}{n}$$

donde m, θ y n son parámetros que dependen del robot. En la ecuación, m se refiere al número de pulsos para cada revolución. θ es el ángulo que se corresponde con el cambio de estado del decodificador de cuadratura y n la relación de transmisión entre el motor y el eje.

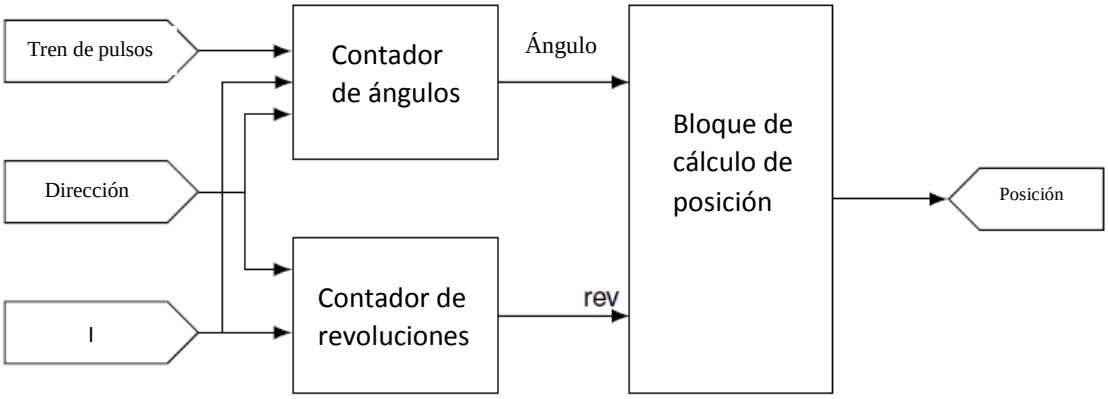


Figura 10: Esquema para el cálculo de la posición.

Por otra parte, para el cálculo de la velocidad se utiliza un tacómetro digital con el fin de mejorar el rendimiento dinámico en el bucle de control ya que, pese a que el uso de los tacómetros analógicos está muy extendido en la industria, los digitales ofrecen una precisión mejorada (error relativo menos del 2%, además, no es necesario ningún conversor A/D o aplicar filtros de ruido analógico.

Existen diferentes técnicas para asegurar una buena precisión y un buen tiempo de respuesta. Uno de ellos es el método del tiempo transcurrido constante [19], el cual mide

el tiempo transcurrido entre un número variable de pulsos k. Para obtener una mayor exactitud, k es actualizada en cada uno de los intervalos de medición. Concretamente, con el fin de obtener la velocidad lo más precisa posible, el número de pulsos, "Tren de pulsos", se cuenta en un periodo fijo de tiempo. Sin embargo, dicha medida sólo nos garantiza la precisión en los intervalos de velocidad bajo y medio. [20] Para el caso de las velocidades altas, el método de recuento de pulsos ofrece un rendimiento mejor. Ambos métodos han sido implementados en lenguaje VHDL.

Método 1: Método del tiempo transcurrido constante: Se basa en la activación de un contador de N- bits por un pulso (contador ascendente), dicho contador se detiene cuando se alcanza el número máximo de pulsos, k.

Se define 'f_b' como la frecuencia de entrada al contador, y C es el número máximo de pulsos que se ha predefinido, la velocidad angular 'ω_{tc}' de un eje se determina por la ecuación:

$$\omega_{tc} = \frac{2\pi f_b}{n}\frac{K}{C}$$

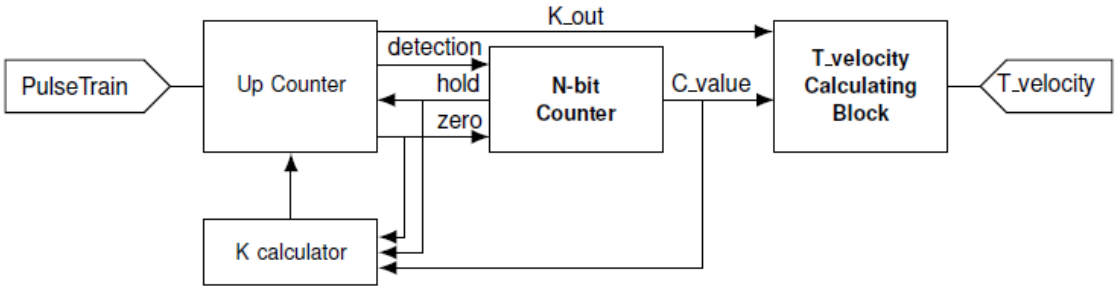


Figura 11 Esquema del tiempo transcurrido constante.

Método 2: Método del contador de pulsos: Para un intervalo de tiempo fijo, 'T_c' se cuentan los pulsos del decodificador y se almacenan en PT_C. La velocidad angular ω_{pc} es:

$$\omega_{pc} = \frac{2\pi f_c}{n}\frac{PT\_C}{m}$$

siendo, $fc = \frac{1}{T_c}$, 'm' el número de pulsos por revolución y 'n' es la relación de transmisión entre el motor y el eje.

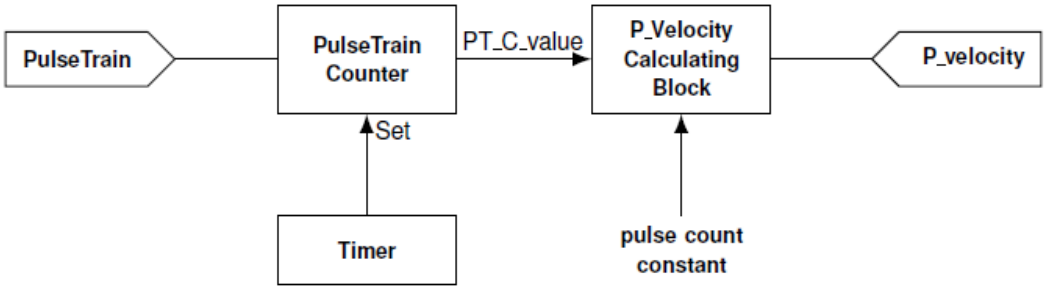


Figura 12 Esquema del método de contador de pulsos para obtener la velocidad.

Finalmente, la aceleración se calcula en base a los pulsos del decodificador. A partir, del método de recuento de pulsos durante un periodo de tiempo 'Tc', donde los pulsos de entrada se cuentan en función de un contador de tren de pulsos y se carga en un registro de desplazamiento la aceleración que viene determinada por la ecuación:

$$a = \frac{2\pi}{nTc^2}\frac{actual - anterior}{m}$$

siendo 'n' la relación de transmisión entre el motor y el eje. Y 'm' es el número de pulsos por revolución.

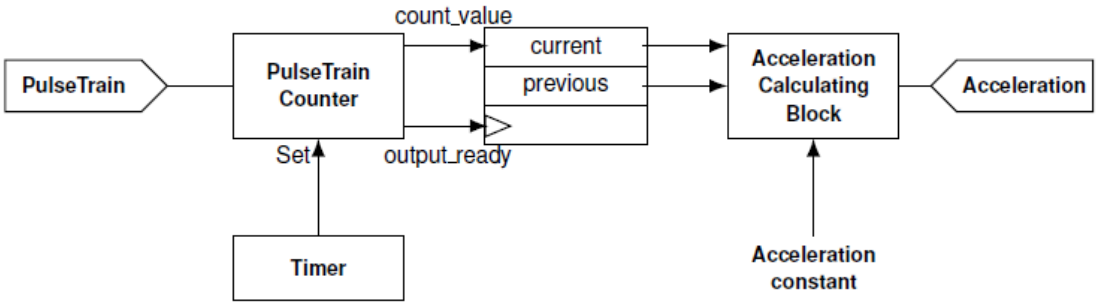


Figura 13: Esquema para el cálculo de la aceleración.

6 Conclusiones

El sistema de visión de un sistema de control visual es el encargado de proporcionar la información para el cálculo de la acción de control. En este proyecto se ha descrito una arquitectura abierta que, embebida sobre una FPGA, es capaz de proporcionar dicha información visual a mayor velocidad de que proporciona cualquier sistema desarrollado sobre un procesador de propósito general como un ordenador.

Para ello, se ha propuesto un cauce segmentado que permite realizar las operaciones necesarias para aplicar la "Ley de control". El desarrollo de la arquitectura propuesta en dicho proyecto permite el desarrollo de nuevos esquemas de control visual embebidos en una FPGA.

Inicialmente no se disponía de la electrónica necesaria para la implementación de la arquitectura propuesta por lo que como trabajo de continuación de este proyecto, se validará la arquitectura mediante la utilización de un patrón de un punto para realizar el guiado del robot multi-articular COOPER utilizando dos de sus tres grados de libertad.

7 Referencias

[1] Pritschow G. Automation technology—on the way to an open system architecture. Robotics and Computer Integrated Manufacturing 1990;7(1/ 2):103–11.

[2] Morales-Vel´azquez, L., de Jesus Romero- Troncoso, R., Osornio-Rios, R. A., Herrera- Ruiz, G., Cabal-Yepez, E., 2010. Openarchitecture system based on a reconfigurable hardware–software multi-agent platform for CNC machines. Journal of Systems Architecture 56, 407–418.

[3] Hassan, S., Anwer, N., Khattak, Z., & Yoon, J. (2010). Open architecture dynamic manipulator design philosophy (DMD). *Robotics and Computer-Integrated Manufacturing*, *26*(2), 156–161. doi:10.1016/j.rcim.2009.07.006

[4] Lalpuriya, K., Prakash, N. R., & Rastogi, N. (2013). FPGA based control system for Robotic Applications, 4(6), 2864–2867.

[5] Shao, X., & Sun, D. (2007). Development of a New Robot Controller Architecture with FPGABased IC Design for Improved High-speed Performance, 3(4), 312–321.

[6] Zhang, L., Slaets, P., & Bruyninckx, H. (2012). An open embedded hardware and software architecture applied to industrial robot control. 2012 IEEE International Conference on Mechatronics and Automation, 1822–1828.

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

[8] Spong, M.W., (1989) "Robot Dynamics and Control", p. 336, John Wiley & Sons, Inc., New York, NY, USA.

[9] Kelly, R., J. Favela, J.M. Ibarra y D. Bassi, (2000) "Asymptotically stable visual servoing of manipulators via neural networks", *Journal of Robotic Systems*, vol. 17, nº 12, pp. 659-669.

[10] Takegaki, M. y S. Arimoto, (1981) "A New Feedback Method for Dynamic Control of Manipulators", *J. of Dyn. Syst., Meas., and Control*, vol. 103, nº 2, pp. 119-125.

[11] Hashimoto, H., T. Kubota, M. Sato y F. Harashima, (1992) "Visual control of robotic manipulator based on neural networks", *Industrial Electronics, IEEE Transactions on*, vol. 39, nº 6, pp. 490-496.

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

[13] Sciavicco, L., B. Siciliano y B. Sciavicco, (2000) "Modelling and Control of Robot Manipulators", p. 402, Springer-Verlag, London, United Kingdom.

[14] http://www.verilog.com/ Visitado en Junio 2014.

[15] Alabdo, A., García, G. J., Pomares, J. y Torres, F. (2014) "Arquitectura abierta de control visual directo sobre FPGA". Jornadas de Automática 2014.Valencia.

[16] http://www.xilinx.com/products/boards-and-kits/EK-K7-KC705-G.htm Visitado en Julio de 2014.

[17] Perea, I., García, G.J., Jara, C.A., Pomares, J., Candelas, F.A., Torres, F., (2011) "COOPER (COupled OPEration Robot): control de una estructura articular RRR acoplada a un manipulador", Actas de las XXXII Jornadas de Automática, Sevilla.

[18] Zhang, L., Slaets, P. y Bruyninckx, H. (2012)"An Open Embedded Hardware and Software Architecture Applied to Industrial Robot Control" Proceedings of 2012 IEEEInternational Conference on Mechatronics and Automation. Chengdu, China.

[19] M. Bonert, “Digital tachometer with fast dynamic response implemented by microprocessor,” *IEEE Transactions on Industrial Applications*, vol. IA-19, pp. 1052–1056, 1983.

[20] E. Galvan, A. Torralba, and L. G. Franquelo, “ASIC implementation of a digital tachometer with high precision in a wide speed range,” *ITIE*, vol. 43, no. 6, pp. 655–660, 1996.

8 Agradecimientos

En primer lugar quisiera darle las gracias a mi familia por su apoyo incondicional, sin el cual no podría haber realizado el máster. Por otra parte, me gustaría agradecer el apoyo recibido de los miembros del grupo Aurova, en especial a Jorge, Carlos y Aiman. Y por último, pero no menos importante, mil gracias a Gabriel, mi tutor, eres todo un MAESTRO.