Universitat d'Alacant
Universidad de Alicante

ARQUITECTURA EMBEBIDA EN FPGA PARA CONTROL VISUAL DINÁMICO BASADO EN IMAGEN DE ROBOTS MANIPULADORES

Aiman Alabdo

Tesis Doctorales

www.eltallerdigital.com

UNIVERSIDAD de ALICANTE

Universitat d'Alacant
Universidad de Alicante

ARQUITECTURA EMBEBIDA EN FPGA PARA CONTROL VISUAL DINÁMICO BASADO EN IMAGEN DE ROBOTS MANIPULADORES

Aiman Alabdo

Tesis doctoral

Alicante, julio 2017

Universitat d'Alacant
Universidad de Alicante
Departamento de Física, Ingeniería de Sistemas y Teoría de la Señal
Escuela Politécnica Superior

ARQUITECTURA EMBEBIDA EN FPGA PARA CONTROL VISUAL DINÁMICO BASADO EN IMAGEN DE ROBOTS MANIPULADORES

Aiman Alabdo

Tesis presentada para aspirar al grado de
DOCTOR POR LA UNIVERSIDAD DE ALICANTE

Doctorado en Automática y Robótica

Dirigida por:
Jorge Pomares Baeza
Gabriel Jesús García Gómez

إلى أميرتي الصغيرة، لورين ...

Agradecimientos

El desarrollo de la presente Tesis Doctoral no hubiera sido posible sin un conjunto de personas a las que, en este momento, al finalizar esta etapa de mi vida, me gustaría dar un especial reconocimiento y agradecer su contribución, de una u otra forma, a este trabajo.

A mi padre y mi madre por confiar en mí y estar conmigo siempre, brindando su apoyo, orientación, cercanía y motivación a seguir estudiando a pesar de los duros tiempos de guerra que están viviendo en mi país Siria desde hace seis años y hasta el momento.

A Dalia, mi mujer, por apoyarme a lo largo de estos cuatro años, y por estar a mi lado y comprender mi manera de ser. Te quiero amor.

A los directores de esta tesis, Jorge y Gabriel, por estar siempre disponibles y aclarar las dudas durante el desarrollo de la tesis, y por sus consejos y críticas constructivas.

A mis compañeros del laboratorio de Automática y Robótica del Departamento de Física, Ingeniería de Sistemas y Teoría de la Señal en la Universidad de Alicante, por compartir sus experiencias y estar siempre dispuestos a echar ese cable cuando se ha necesitado.

A los profesores del grupo AUROVA por su ayuda y sus ánimos para llegar a cumplir este trabajo.

A Foro, mi gran amiga, por darme siempre fuerza y cariño desde mi primer día en España y hasta ahora.

A Ángela y Alejandro, mi queridísima familia en Alicante.

Finalmente, me gustaría dar las gracias a las personas más importantes de mi vida, a las que han disfrutado y sufrido conmigo la decisión de dedicarme en exclusiva a realizar mis estudios desde los 18 años, lejos de ellos: a toda mi familia, en especial a mi hermana Eyman quien se dedicó a enseñarme todo desde niño y me motivaba siempre para llegar a este punto.

A todos los que, por despiste (disculpadme), puedo haber olvidado.

Aiman Alabdo

Resumen

En esta Tesis Doctoral se presenta una arquitectura para el control visual de robots manipuladores embebida en FPGA. El objetivo fundamental de esta arquitectura es obtener una plataforma hardware y software en la que implementar controladores visuales dinámicos. Además, en la Tesis se presentan distintas contribuciones entre las que se encuentra tanto la propia arquitectura como los distintos controladores implementados y las optimizaciones de los mismos.

La mayoría de los sistemas de control visual basados en imagen implementados hasta la actualidad son controladores visuales indirectos en los que la acción de control está formada por las velocidades articulares o en el espacio de la tarea a aplicar al robot para conseguir su posicionado respecto a un objeto observado. El control directo de los motores de cada articulación del robot se deja en este caso en manos del controlador interno del robot, que traducirá esas velocidades en pares articulares. En esta Tesis se trata fundamentalmente el caso de sistemas de control visual directo basados en imagen para el seguimiento de trayectorias. En este caso, la acción de control es ya directamente un vector de pares articulares, con el propósito de seguir una trayectoria previamente especificada en el espacio de la imagen. Un aspecto fundamental a la hora de mejorar el desempeño de este tipo de controladores es reducir su latencia y, para abordar este objetivo, se ha realizado el diseño, construcción, implementación y programación de la arquitectura hardware y software necesaria para dar soporte a los controladores propuestos.

En la Tesis se describe en detalle una arquitectura embebida reconfigurable para el control visual dinámico de robots manipuladores basada en FPGA. Esta arquitectura puede ser fácilmente adaptada para el control de cualquier robot articular sin más que modificar ciertos módulos dependientes del hardware. La arquitectura propuesta es modular y flexible para poder adaptarse a posibles cambios que puedan producirse como consecuencia de la incorporación o modificación del driver de control, o incluso a cambios en la configuración del sistema de adquisición de datos o su control. Se ha prestado

especial atención a la optimización e implementación paralela en la FPGA tanto de los controladores como los sistemas de visión artificial utilizados en la realimentación del bucle de control. Se ha optimizado cada uno de los cálculos necesarios para implementar un sistema de control visual directo basado en imagen. Se ha estudiado su paralelización y su implementación basada en FPGA Además, respecto a los sistemas de visión artificial, se propone el diseño de un cauce segmentado, que permite ejecutar en tiempo real el flujo de datos proporcionado por el digitalizador. De esta manera, la FPGA realiza el procesamiento tal como van llegando los píxeles para segmentar la imagen y etiquetar los objetos de la escena para posteriormente obtener su centroide que constituirán las características visuales empleadas por los controladores visuales directos propuestos.

En estos grandes rasgos se encuadra la Tesis Doctoral, sin embargo, no sólo se ha pretendido aportar nuevos algoritmos para evitar las limitaciones de los existentes, sino que se ha evaluado el impacto de los mismos para su implantación en entornos reales. Así, se ha realizado la implementación de los distintos controladores empleando la arquitectura software y hardware propuesta para el control de dos robots. Un robot industrial comercial de 7 grados de libertad: Mitsubishi PA10, y otro robot de 3 grados de libertad cuyo diseño e implementación se ha realizado en el grupo de investigación en el que se ha desarrollado la Tesis.

Abstract

In this PhD Thesis an FPGA embedded architecture for visual control of robot manipulators is presented. The fundamental objective of this architecture is to obtain a hardware and software architecture for implementing dynamic visual servoing systems. In addition, this Thesis presents different contributions like the architecture itself as well as the different implemented controllers and the performed optimizations.

Most of the image-based visual servoing systems implemented to date are indirect visual controllers in which the control action are joint or end-effector velocities to be applied to the robot to achieve a given desired location with respect to an observed object. The direct control of the motors for each joint of the robot is performed by the internal controller of the robot, which translates these velocities into joint torques. This Thesis mainly addresses the direct image-based visual servoing systems for trajectory tracking. In this case, in order to follow a given trajectory previously specified in the image space, the control action is defined as a vector of joint torques. A fundamental aspect for improving the performance of this type of controllers is to reduce their latency. Designing, constructing, implementing and programming a more adequate hardware and software architecture becomes necessary to support the proposed visual controllers. All these different steps are described throughout the Thesis.

In the Thesis, an embedded FPGA-based reconfigurable architecture for dynamic visual servoing of robot manipulators is described. This architecture can be easily adapted for controlling of any joint robot by simply modifying certain modules that are hardware dependents. The proposed architecture is modular and can be adapted to possible changes that may occur as a consequence of the incorporation or modification of a control driver, or even changes in the configuration of the data acquisition system or its control. This Thesis describes in detail the optimization and parallel implementation in the FPGA of both the controllers and the computer vision systems used in the feedback of the control loop. Each of the computations required to implement a direct image-based visual servoing system has been optimized. Its FPGA-based parallelization and implementation

are studied. In addition, regarding to the computer vision system, the design of a pipeline strategy is proposed, such strategy allows for processing data flow provided by the frame grabber at runtime. In this way, the FPGA performs the processing as pixels arrive for segmenting the image and labeling the objects in the scene for obtaining its centroid. These centroids are the visual features employed by the proposed direct visual controllers for performing the robot guidance.

These are the main objectives of the PhD Thesis. However, the contribution of new algorithms for avoiding the limitations of existing ones is not the only work performed in the Thesis. Also, the implementation of the proposed algorithms in real environments is evaluated. Thus, the implementation of the direct visual servoing systems is performed using the proposed FPGA software and hardware architecture for the control of two different robots. A commercial industrial 7 degrees of freedom robot: Mitsubishi PA10, and another 3 degrees of freedom robot whose design and implementation has been developed in the research group where the Thesis is written.

1 Introducción

En este Capítulo se comentan las principales motivaciones que han propiciado la realización de esta Tesis Doctoral, enmarcando las investigaciones realizadas dentro de tres grandes proyectos. También se presentan en este Capítulo las aportaciones realizadas en el ámbito de los sistemas de control visual embebidos en FPGA que se detallarán a lo largo de la Tesis. Por último, se describirá cómo se ha estructurado el documento, indicando qué temas se abordan en cada Capítulo.

1.1 Motivación

Actualmente, los sistemas de visión artificial están integrados en un gran número de aplicaciones industriales, ya sea como método de detección de defectos en líneas de producción, para el reconocimiento de objetos o patrones, o como método para el escaneado de superficies, por ejemplo. Dentro de estas aplicaciones industriales cabe destacar los sistemas de control visual para el guiado de manipuladores robóticos. El principal objetivo de los sistemas de control visual es la utilización de visión artificial como herramienta para el guiado de los robots. Estos sistemas de control se realimentan con la información captada por una o varias cámaras, de forma que dicho controlador se encarga de establecer las acciones de control oportunas para guiar al manipulador en su tarea. Como se describirá a lo largo de la presente Tesis, los sistemas de control visual están ampliamente extendidos, ya no sólo en los laboratorios de investigación, sino también en un amplio rango de aplicaciones que va desde la robótica industrial a la de servicios. Uno de los aspectos que ha propiciado el auge de los sistemas de control visual, su difusión, y la extensión de sus aplicaciones, ha sido la capacidad de captura y procesamiento de las cámaras de visión actuales, así como de los equipos empleados para su procesamiento. Esto ha permitido procesar mayor cantidad de información en menor tiempo, evitando los posibles retardos existentes y guiando al robot de una forma flexible.

A pesar de los esfuerzos realizados en la última década con el objetivo de mejorar distintos aspectos en el desempeño de estos sistemas de control visual, aún en la actualidad existe un gran número de líneas de investigación abiertas dentro de las cuales cabe destacar la implementación embebida de los sistemas de control visual, así como optimizar sus tiempos de procesamiento, con el propósito de conseguir su ejecución en tiempo real. Como se ha indicado anteriormente, la utilización de un sistema de control visual para el guiado de manipuladores requiere la utilización de visión artificial en la realimentación del sistema de control, por lo tanto, es necesario procesar gran cantidad de información en cada iteración del bucle de control. A menudo, este procesamiento constituye el principal retardo en estos sistemas de control. Este tiempo de procesamiento puede reducirse empleando, entre otros métodos, procesamiento paralelo. Por otro lado, una FPGA (del inglés *Field Programmable Gate Array*) es un hardware programable que contiene bloques de lógica cuya interconexión y funcionalidad puede ser configurada 'in situ' mediante un lenguaje de descripción especializado. A lo largo de la Tesis se describirá la utilización de algoritmos implementados sobre FPGA con el propósito de no sólo implementar los procesamientos necesarios en las imágenes capturadas, sino también cada uno de los elementos implicados en las tareas de control visual.

Dentro de los sistemas de control visual cabe destacar los denominados sistemas de control visual directos. Estos controladores se distinguen de los indirectos o cinemáticos en que tienen en cuenta las características dinámicas del robot controlado. Así, la acción de control de los sistemas de control visual directo se compone de los pares articulares a aplicar al manipulador guiado. A lo largo de la Tesis se realizará la formulación de un nuevo sistema de control visual para el seguimiento de trayectorias, así como su implementación optimizada sobre FPGA.

En la mayoría de las ocasiones, a la hora de implementar controladores visuales directos se realiza un desarrollo e implementación 'ad hoc' para un robot específico. Es decir, la implementación realizada es adecuada para el manipulador considerado, pero difícilmente puede ser extendida a otro manipulador con diferentes características cinemáticas y dinámicas. En la presente Tesis se presenta una arquitectura embebida

reconfigurable para el control visual directo de robots manipuladores basada en FPGAs. Esta arquitectura puede ser fácilmente adaptada para el control de cualquier robot articular. Tal y como se describirá a lo largo de esta Tesis, la arquitectura propuesta será modular y flexible para poder adaptarse a posibles cambios que puedan producirse como consecuencia de la incorporación o modificación del driver de control, o incluso a cambios en la configuración del sistema de adquisición de datos.

En estos grandes rasgos se encuadra la Tesis Doctoral, sin embargo, no sólo se ha pretendido aportar nuevos algoritmos para evitar las limitaciones de los existentes, sino que se ha evaluado el impacto de los mismos para su implantación en entornos reales. Así, se ha realizado la implementación de los distintos controladores empleando la arquitectura software y hardware propuesta para el control de dos robots. Un robot industrial comercial de 7 grados de libertad: Mitsubishi PA10, y otro robot de 3 grados de libertad cuyo diseño e implementación se ha realizado en el grupo de investigación en el que se ha desarrollado la Tesis.

1.2 Marco de la Tesis

El primero de los proyectos en los que se enmarca la Tesis es el proyecto de investigación "Manipulación diestra de objetos rígidos y elásticos con guiado mediante control visual-táctil-fuerza", financiado por el Ministerio de Economía y Competitividad (DPI2012-32390) y los fondos FEDER. La finalidad del proyecto fue la investigación de técnicas de control visual y de manipulación-agarre orientadas a su implementación conjunta para aplicarlas a tareas de robótica de servicios o industrial. Así, se pretendió que las soluciones adoptadas puedan ser empleadas satisfactoriamente tanto en entornos industriales donde se requiera la manipulación de sustancias y/o productos peligrosos para el ser humano, como en entornos cotidianos en los que haya limitaciones posturales o de fuerza. En algunas de estas tareas, no siempre se dispone de un conocimiento previo exhaustivo del objeto o sustancia que se desea manipular. Por lo tanto, se precisa de dos tipos de sistemas. Primero, un sistema de percepción visual que permita reconocer el objeto que se pretende manipular y que permita llevar a cabo un control supervisado de cualquier trayectoria de movimiento. Y, segundo, un sistema robótico articular que

permita realizar, conjuntamente, tareas de agarre y manipulación, imitando la manipulabilidad y precisión articular del ser humano en actividades como el agarre, contacto y percepción táctil de objetos (rugosidad y forma de superficies), o la coordinación táctil para el giro de éstos. La aportación al proyecto de la investigación realizada en la Tesis ha sido en el desarrollo de sistemas de control visual basados en imagen para el guiado de los manipuladores. Se ha trabajado fundamentalmente en la optimización de los mismos con el objetivo final de implementar leyes de control visual directo.

Otro proyecto en el que se ha enmarcado la Tesis ha sido "Control visual embebido de robots manipuladores utilizando hardware reconfigurable FPGA" (GRE12-17), proyecto financiado por el Vicerrectorado de Investigación, Desarrollo e Innovación de la Universidad de Alicante. La finalidad del proyecto fue la investigación en técnicas de control visual embebido sobre FPGA. Para ello se estudiaron las diferentes posibilidades de conexión de una cámara rápida a la FPGA. El empleo de una FPGA para gestionar el procesamiento de la imagen y el cálculo de la ley de control requiere una conexión fluida con la cámara. Los sistemas de control visual directos requieren componentes de alta velocidad de respuesta, que mejoren la precisión del controlador al disminuir el tiempo de iteración del bucle de control. En este proyecto se planteó, no sólo la implementación del controlador directo sobre una FPGA para acelerar el procesamiento de imágenes, sino también para añadir determinismo en el cálculo de la acción de control, con lo que se obtiene un tiempo exacto de iteración. Por lo tanto, la Tesis permitió abordar dos de los objetivos fundamentales de este proyecto como es investigar en sistemas de control embebidos sobre FPGA, así como desarrollar un esquema de control visual directo sobre FPGA.

Por último, las investigaciones y desarrollos llevados a cabo en la Tesis se están aplicando al proyecto financiado por el Ministerio de Economía y Competitividad y fondos FEDER "Sistema robótico multisensorial con manipulación dual para tareas asistenciales humano-robot" (DPI2015-68087-R). En este proyecto se está realizando el diseño, construcción, control y programación de un torso robótico con el objetivo de servir

de asistencia en tareas de manipulación a personas que presentan algún tipo de discapacidad motora como ayuda en sus tareas cotidianas. Para ello, se ha diseñado un torso con dos brazos manipuladores dotados de sendas manos robóticas. La colaboración robot-humano en un mismo entorno para proporcionar un servicio o asistencia demanda sistemas robóticos sensibles y dotados de altas funcionalidades sensoriales en cuanto a seguridad. Además, deben adaptarse al entorno para dar un servicio y/o asistencia, y reaccionar ante determinadas situaciones, como colisiones o cambios inesperados en el entorno. Este proyecto pretende desarrollar un sistema robótico multisensorial con estas características. Así, el robot ha de ser capaz de adaptarse a entornos dinámicos, cambiantes, no estructurados y en interacción robot-humano. Por este motivo se ha incluido en el sistema de manipulación robótico sistemas sensoriales basados en visión, fuerza y táctil que permitan integrar en el robot la capacidad perceptiva necesaria para manipular objetos no conocidos, o que no se encuentran modelados con anterioridad. La Tesis se ha centrado en la aplicación de la arquitectura propuesta de control visual para realizar el guiado de los manipuladores durante su tarea.

1.3 Objetivos de la Tesis

Para finalizar este capítulo introductorio, se van a describir los objetivos y principales aportaciones llevadas a cabo con la presente Tesis. El objetivo general de la Tesis es el desarrollo de una arquitectura basada en FPGA para la implementación de nuevas leyes de control visual directo. Este objetivo general se puede desglosar en los siguientes puntos:

- Diseño e implementación de una arquitectura hardware optimizada para sistemas de control visual directo. A pesar de no existir previamente arquitecturas similares basadas en FPGA para la implementación de sistemas de control visual directo basados en imagen, se ha realizado un estudio de sistemas de control con características similares [GarciaGJ-2014]. Este estudio ha sido el punto de partida para la arquitectura presentada en el Capítulo 3.

- Diseño e implementación de una arquitectura software semiabierta basada en FPGA para la implementación de sistemas de control visual directo. Esta arquitectura, conjuntamente con la arquitectura hardware permiten que pueda aplicarse los controladores a distintos robots aprovechando la reconfigurabilidad de las FPGAs.
- Diseño e implementación de sistemas de visión para control visual basados en FPGA. Haciendo uso de la arquitectura hardware y software anteriormente comentadas se han propuesto y optimizado algoritmos de visión basados en FPGA para ser utilizados como realimentación del bucle de control de estos sistemas.
- Formulación de leyes de control visual directo basadas en imagen para el seguimiento de trayectorias. Se han diseñado leyes de control visual directo entre las que cabe destacar un controlador basado en dinámica inversa para el seguimiento de trayectorias especificada en el espacio imagen.
- Optimización e implementación paralela basada en FPGA de los sistemas de control visual directo. Se ha optimizado cada uno de los cálculos necesarios para implementar un sistema de control visual directo basado en imagen. Se ha estudiado su paralelización y su implementación basada en FPGA.

1.4 Estructura de la Tesis

Una vez descrita la motivación y marco de la Tesis, a continuación, se describe brevemente la manera en que se ha estructurado la presente Tesis doctoral (en la Figura 1.1 se muestra un esquema con los distintos capítulos). Tras este primer capítulo introductorio se describe el marco teórico en el Capítulo 2. Este marco teórico incluye un estado del arte en los dos grandes campos en los que se centran las principales aportaciones de la Tesis: implementación hardware basada en FPGA de controladores y sistemas de control visual directo basados en imagen. Este marco teórico permitirá centrar progresivamente las aportaciones en comparación con el estado del arte actual.

Los tres siguientes capítulos constituyen el núcleo principal de la Tesis. El tercer capítulo detalla la arquitectura hardware propuesta basada en FPGA para la implementación de controladores visuales para manipuladores. Dentro de esta arquitectura destacan dos grandes bloques: implementación basada en FPGA de sistemas de visión artificial para control visual, e implementación basada en FPGA de controladores visuales directos. Estos dos grandes bloques constituyen los capítulos 4 y 5 respectivamente.

Finaliza la Tesis con un capítulo de conclusión en el que se describen las aportaciones, conclusiones y trabajos futuros.

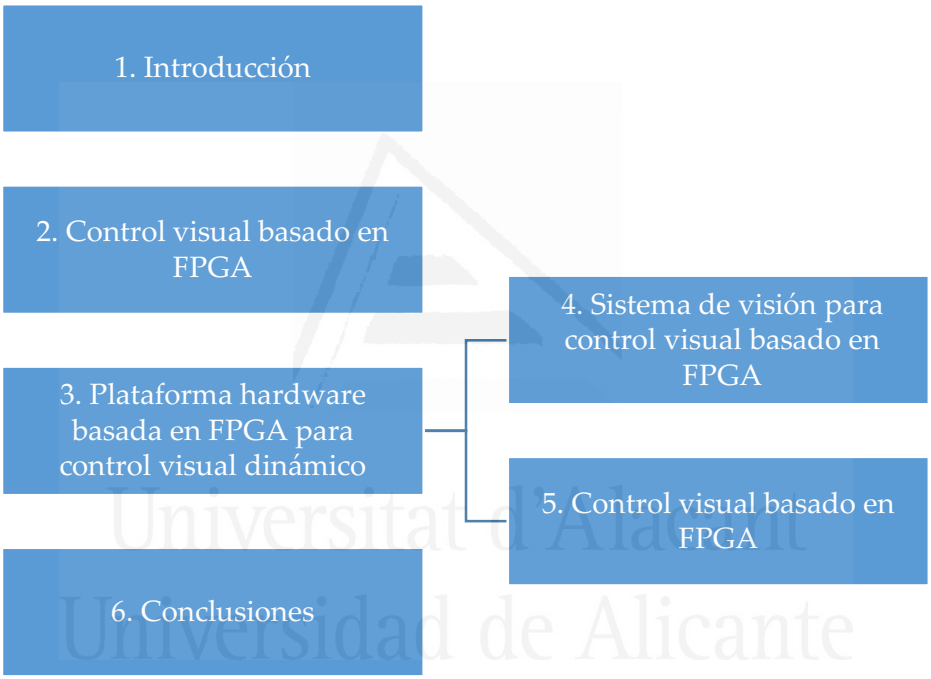


Figura 1.1 Estructura de la Tesis

2 Control visual basado en FPGA

Este capítulo presenta una revisión del estado del arte sobre controladores embebidos en FPGA, profundizando en los conceptos más relacionados con la Tesis e indicando aquéllos en los que se centrarán las aportaciones realizadas. Para ello, comienza el capítulo con un primer apartado introductorio indicando las ventajas de las FPGA frente a otros sistemas de implementación hardware y software de controladores. Posteriormente, se describirán los principales desarrollos encontrados en la literatura de controladores embebidos en FPGA, centrando la atención en los controladores de robots y más particularmente en los controladores visuales de robots. En el tercer apartado se describirán con mayor profundidad los fundamentos de los sistemas de control visual y cuál va a ser su utilización dentro de la Tesis. En este apartado se indicarán las diferentes tipologías que presentan los sistemas de control visual atendiendo a la ubicación del sistema de visión. Se describirá también la principal clasificación realizada comúnmente en los sistemas de control visual: basados en posición y basados en imagen, así como la otra clasificación importante que separa los sistemas de control visual en función del diseño del lazo de control: control visual indirecto y control visual directo. En el cuarto apartado se muestra una implementación de un sistema de control visual directo basado en imagen, con el objetivo de identificar aquellos aspectos en los que se centrarán las aportaciones llevadas a cabo a lo largo de la Tesis. Finaliza el capítulo con un resumen de las aportaciones realizadas en la Tesis dentro del ámbito de los sistemas de control visual.

2.1 Introducción

Habitualmente, los sistemas de control de un robot se suelen implementar en procesadores de propósito general (GPP de sus siglas en inglés *General Purpose Processor*) o en circuitos de propósito específico (ASIC de sus siglas en inglés *Application-Specific Integrated Circuit*). La elección de uno u otro hardware presenta a los ingenieros

alternativas situadas en polos totalmente opuestos [Wang-2013]. Evidentemente, un ASIC está diseñado para una tarea específica y, por lo tanto, su lógica está optimizada para el funcionamiento del sistema integrado en ese circuito. La principal desventaja de los ASICs frente a los GPPs es que los primeros pierden la flexibilidad de los segundos para modificar el diseño inicial del controlador. Es decir, en aplicaciones que exigen una eficiencia de alto rendimiento, un ASIC tiene una eficiencia extremadamente alta, pero ofrece pocas opciones si hay algún cambio de especificación. En el otro lado, los GPPs son completamente programables, pero tienen una eficiencia de rendimiento menor. Así, los ingenieros deben elegir entre eficiencia de rendimiento y reprogramación. Cuando se consideran los requisitos de potencia como la principal limitación, la eficiencia es la elección. Por lo tanto, los ASICs serían la principal prioridad, abandonando la posibilidad de la reprogramación. Esto significa que se admitirá una sola especificación del controlador, y que se necesita un ASIC distinto para cada nueva especificación, con el coste económico y temporal que esto supone.

Por otro lado, las tendencias de evolución de los procesadores GPP parecen indicar una mejora de la eficiencia de rendimiento. Históricamente, la velocidad de los procesadores ha aumentado año tras año, impulsada por mayores densidades de transistores. Además, toda una serie de técnicas avanzadas como cauces segmentados extendidos, ejecución fuera de orden, programación dinámica y almacenamiento en caché de memoria multinivel tratan de abordar el problema. Sin embargo, con densidades que se aproximan a los niveles atómicos, hoy en día estas mejoras producen rendimientos poco apreciables, ya que la eficiencia mejora más lentamente que la densidad [Agarwal-2000], [Debenidictis-2017].

Una FPGA es un hardware programable que contiene bloques de lógica cuya interconexión y funcionalidad puede ser configurada 'in situ' mediante un lenguaje de descripción especializado [Freeman-1989]. Una FPGA proporciona, por tanto, la posibilidad de reconfiguración de los sistemas de control programados sobre un GPP, pero también proporciona el alto rendimiento de una programación hardware específica que se obtiene con los ASICs [Kuon-2007].

Las capacidades actuales de las FPGAs permiten no sólo implementar circuitos combinacionales y secuenciales simples, sino también disponer de procesadores software de alto nivel [Tong-2006]. El uso de procesadores integrados tiene muchas ventajas a tener en cuenta por el diseñador de los controladores, desde facilidad de personalización, pasando por la mitigación de la obsolescencia, reducción de componentes y costes hasta aceleración de hardware. Los procesadores embebidos en una FPGA utilizan los elementos lógicos de ésta para construir unidades de memoria interna, buses de datos y control, periféricos internos y externos y controladores de memoria. Tanto Xilinx como Altera (los dos líderes del mercado en la industria FPGA) proporcionan dispositivos FPGA que incorporan procesadores físicos construidos dentro del chip FPGA. Este tipo de procesadores se llama procesador 'hardware' [Rodriguez-Andina-2007]. Este es el caso del PowerPCTM 405, que se puede encontrar en los Virtex-4 de Xilinx, y el ARM922TTM, dentro de los dispositivos Excalibur de Altera. Por otro lado, los procesadores 'software' son microprocesadores cuya arquitectura se implementa totalmente usando un lenguaje de descripción hardware (HDL, del inglés *Hardware Description Language*). La ventaja de usar este tipo de procesador es que el ingeniero puede implementar el número exacto de procesadores software requerido por la aplicación, ofreciendo gran flexibilidad al diseñador para realizar modificaciones o adaptarse al controlador específico que quiere implementar. Los procesadores software más famosos son: el LEON3 de Aeroflex Gaisler [Gaisler-2007], que es un modelo VHDL (del inglés *Very High Speed Integrated Circuit and HDL*) de un procesador de 32 bits compatible con la arquitectura SPARC V8; el procesador software Nios II [Altera-2009], que es una arquitectura de procesador embebido de 32 bits diseñada específicamente para los dispositivos FPGA de Altera; y el procesador software MicroBlaze de Xilinx [Xilinx-2004]. Este último procesador es un procesador software de arquitectura RISC (del inglés *Reduced Instruction Set Computer*) Harvard de 32 bits con un rico conjunto de instrucciones optimizado para aplicaciones embebidas.

Los recursos de hardware que se implementan en las FPGAs son muy diferentes dependiendo del fabricante y de las FPGAs específicas. Sin embargo, un gran número de dispositivos incluyen componentes que los hacen adecuados para ser aplicados en

sistemas de controladores. Este es el caso de los procesadores descritos anteriormente y de los transceptores. Un transceptor es un SerDes (serializador / deserializador) capaz de operar a velocidades de bits en serie de hasta 28.05 Gigabit / segundo en FPGAs actuales como los dispositivos FPGA Stratix V de Altera y Virtex-HT HTGA de Xilinx. Se utilizan cada vez más para las comunicaciones de datos porque pueden recorrer distancias más largas, utilizar menos cables y, por lo tanto, tener costes más bajos que las interfaces paralelas con un rendimiento de datos equivalente. La mayoría de las FPGAs pueden proporcionar GPIOs (del inglés *General Purpose Input/Output*) configurables para permitir que una amplia gama de dispositivos sean conectados y operados a diferentes niveles de voltaje sin la necesidad de usar adaptadores de interfaz o convertidores de voltaje, simplificando significativamente el diseño y reduciendo costos. Por ejemplo, la FPGA Spartan-3 de Xilinx proporciona varios estándares de bancos de E / S como LVCMOS, LVTTL, GTL, HSTL, PCI, SSTL, LDT, LVDS, RSDS y LVPECL que pueden operar a diferentes niveles de voltaje de 1.2V a 3.3V.

Algunas FPGAs incorporan una gran cantidad de bloques aritméticos, que pueden ser bloques de baja complejidad tales como multiplicadores simples, o pueden ser relativamente más complejos como las unidades DSP (del inglés *Digital Signal Processing*) que consisten en la combinación de varios componentes como multiplicadores, sumadores, acumuladores, registros, etc. Una unidad DSP acelera significativamente el rendimiento de la FPGA y permite lograr una mayor productividad y flexibilidad, disminuyendo el coste y el consumo de energía. Por ejemplo, cada Stratix II y Stratix II GX (de Altera) tiene de dos a cuatro columnas de bloques DSP que implementan de manera eficiente las funciones de multiplicación, multiplicación-acumulación y multiplicación-suma. El número de bloques DSP por columna y el número de columnas disponibles depende del dispositivo. Por ejemplo, el dispositivo EP2S180 de la familia Stratix II tiene 96 bloques DSP, así como 769 multiplicadores de 9x9, 384 de 18x18 y 96 de 36x36. Otro aspecto que mejora el rendimiento de las FPGAs es el uso de las memorias internas. Las memorias internas ofrecen una velocidad relativamente alta en comparación con las memorias externas. Las FPGAs actuales contienen gran cantidad de bloques de memoria interna, por ejemplo, hasta 34 Mb de memoria interna en los dispositivos Virtex-6 de

Xilinx. Se pueden encontrar otros tipos de memoria, como RAM (del inglés *Random Access Memory*), ROM (del inglés *Read Only Memory*) o registros de desplazamiento. Además, el diseñador puede implementar otras estructuras de memoria como FIFO (del inglés *First In First Out*).

A final de la primera década de 2000 ya existía una gran variedad de diseños de controladores embebidos en FPGA [Mommasson-2007]. Sin embargo, es en esta segunda década cuando se encuentran en la literatura más trabajos relacionados con controladores embebidos en FPGA. Se pueden encontrar controladores en aplicaciones tales como robótica, electrónica de potencia y motores. En el siguiente apartado se describirá el estado del arte de controladores embebidos en FPGA.

2.2 Controladores en FPGA

El uso de FPGAs en sistemas de control industrial es de gran interés debido al creciente nivel de requisitos de los controladores [Wen-Hong-2013]. Una FPGA permite implementar una arquitectura paralela dedicada que se puede adaptar en tiempo de ejecución a las necesidades del sistema. Las FPGAs ya han sido utilizadas con éxito en diferentes sistemas de control, ya sea en la implementación de controladores de lógica difusa [HuangHC-2016b], [Sánchez-Solano-2013], [Sulaiman-2009], controladores de movimiento [Gamazo-2010], [Pérez-Peña-2013], redes neuronales [Gadea-2000], [De Garis-2002], [Ortigosa-2011], [Rosado-2012], [Muthuramalingam-2008], [HuangHC-2016a], control de motores asíncronos [Calmon-2002], control del convertidor de potencia [SánchezPM-2013], sistemas mecatrónicos [MacCleery-2008], etc.

La implementación en hardware de un sistema de control puede mejorar sustancialmente su rendimiento. Sin embargo, los recursos FPGA son limitados, y los algoritmos de los sistemas de control deben ser refinados. Este último aspecto es ampliamente investigado por la comunidad científica, que ha dedicado un gran esfuerzo a optimizar los recursos de las FPGAs en la implementación de los algoritmos de los sistemas de control. Por ejemplo, en [Sánchez-Solano-2013] se propone un método de diseño basado en modelos para la síntesis de controladores difusos embebidos para el

desarrollo conjunto de componentes de hardware y software. Ya en trabajos anteriores, se describían controladores difusos básicos embebidos en FPGA [Sánchez-Solano-2007]. Aunque es posible implementar controladores basados en sensores FPGA con aritmética de punto flotante [Díaz-2006], los recursos necesarios no están optimizados con respecto al punto fijo. CORDIC (del inglés *Coordinate Rotation Digital Computer*) es un algoritmo usado para aproximar iterativamente algunas funciones usando sumadores/sustractores y desplazadores [Volder-1959]. Este enfoque ha sido utilizado por varios autores con el fin de refinar y optimizar un sistema de control embebido en una FPGA [Bravo-2006]. En consecuencia, cuando los sistemas de control deben desarrollarse sobre una FPGA, debe alcanzarse un compromiso entre las prestaciones de control y la complejidad de la arquitectura de hardware. En [Gürsoy-2016] se propone el uso de una FPGA para implementar un esquema de control PID y un esquema de control SMC (del inglés *Sliding Mode Control*) para robots manipuladores. Es habitual encontrar trabajos como éste, donde se implementa el controlador PID en la FPGA mientras que otra parte de control del sistema se programa sobre un DSP [Jung-2007].

El uso de las FPGAs también permite reducir los retardos en la retroalimentación de los sistemas de control. La alta demanda de intercambio de datos puede aprovecharse de la cada vez mayor densidad de los chips en las FPGAs [Moreno-2010]. Es habitual encontrar aplicaciones donde se requiere no sólo capturar la información del sensor en la retroalimentación, sino también procesar dicha información con el fin de obtener los datos necesarios para ser comparados con la referencia del sistema. Dentro de este campo de investigación, es posible mencionar el trabajo descrito en [Osuna-2012], donde se propone una infraestructura de monitorización basada en FPGA. En [Lorenz-2013], se presenta un sistema de visión por computador, basado de nuevo en FPGA, para la medición del nivel de líquido en una aplicación de destilación de membrana. Otro sistema de monitorización se presenta en [Lopez-Buedo-2002], [Lopez-Buedo-2004]. En este caso, se emplean sensores térmicos y se pueden utilizar para detectar, por ejemplo, si un dispositivo dado disipa potencia excesiva o no funciona correctamente.

No hay que olvidar una de las principales ventajas de los sistemas de control embebidos en una FPGA: la reconfigurabilidad en tiempo real de estos dispositivos. En [Ivchenko-2015] se describe un sistema de control de un vehículo espacial autónomo, que implementa su controlador en una FPGA. Gracias a ésta, el *rover* es capaz de adaptarse a la imprevisibilidad del entorno externo en circunstancias en las que la teleoperación es inviable debido a los altísimos retardos.

En la actualidad, las FPGAs se están utilizando no sólo para implementar sistemas clásicos de control, sino también diferentes tipos de sistemas de control, como el control predictivo, sistemas difusos o redes neuronales. El control predictivo es una estrategia de control bien establecida que se está utilizando en un creciente conjunto de áreas de aplicación. La naturaleza paralela de estos controladores se ajusta perfectamente a la arquitectura de las FPGAs [Yanga-2012], y se pueden encontrar diferentes implementaciones en la literatura para operar sobre cualquier variable (temperatura, velocidad, presión, etc.). En [SánchezPM-2013] se describe el uso de estos controladores y su implementación óptima en una FPGA para su aplicación en convertidores de potencia. Otro tipo de sistema de control que se implementa con éxito en FPGAs, es el control difuso [Li-2003], [Sulaiman-2009]. Estos sistemas no requieren un modelado complejo de la planta y la estrategia de control se define mediante el uso de reglas lingüísticas que pueden implementarse utilizando una arquitectura FPGA. La alta carga computacional de los algoritmos difusos puede ser procesada usando la arquitectura FPGA paralela para lograr la precisión deseada en tiempo real. Estos sistemas se han aplicado a multitud de sistemas robóticos, en [Li-2003] se describen controladores difusos embebidos en FPGA para el control de un vehículo robótico autónomo capaz de emular la conducción humana en tareas como el aparcamiento en línea o en batería, o la circulación en túnel. Éste es uno de los primeros trabajos en los que se utiliza una FPGA para implementar un controlador robótico. Ya en 2009, se presenta un controlador robusto adaptativo embebido en FPGA para el seguimiento de trayectorias y la estabilización de una plataforma móvil omnidireccional con variaciones de parámetros e incertidumbre causada por fricción y deslizamiento [HuangHC-2009]. Desde la implementación sobre FPGA de sistemas de control difuso básicos, en la actualidad se puede encontrar en la literatura una gran

cantidad de controladores difusos embebidos modificados. En [HuangSJ-2015] se emplea un controlador difuso de modo deslizante para diseñar el sistema de control de movimiento de una plataforma con dos brazos robóticos. Aunque no se describe en profundidad cómo se implementa el controlador en la FPGA. Recientemente, en [HuangHC-2016b] se describe una metodología de fusión del algoritmo de murciélago modificado y el modelo dinámico para el control difuso en línea auto-adaptativo de robots móviles autónomos en una FPGA. En [Sánchez-Solano-2013], se propone un enfoque basado en modelos para implementar controladores difusos en una FPGA. Otras implementaciones específicas de estos controladores se pueden encontrar en aplicaciones como la automoción [Alvarez-2006], la educación [Alcantara-2007], o incluso el control de un pez robótico para el seguimiento de otros peces o para evitar obstáculos [LeePJ-2012]. En [Masmoudi-2017] se describe una aproximación para el diseño e implementación de un controlador PI de lógica difusa para el sistema de navegación omnidireccional de un robot utilizando una FPGA. En primer lugar, se define el modelo cinemático del sistema de robot y luego se diseña, simula y optimiza el sistema de navegación del controlador utilizando las plataformas MATLAB y Robotino Sim, para finalmente mostrar la implementación hardware con recursos FPGA.

Las capacidades crecientes de las FPGAs han abierto una nueva línea de investigación que trata de obtener nuevas metodologías para escalar controladores o arquitecturas inteligentes en sistemas embebidos basados en FPGAs [Kassas-2011]. Esto puede lograrse mediante la implementación de redes neuronales [Zhu-2003]. Por ejemplo, los trabajos de De Garis et al. describen distintas evoluciones de algoritmos genéticos embebidos en una FPGA para controlar los comportamientos de una mascota robótica [De Garis-2000a], [De Garis-2000b], [De Garis-2001], [De Garis-2002]. En [Jimenez-Fernandez-2009] se propone un controlador de robot móvil con dos motores de corriente continua inspirado en redes neuronales. Una línea de investigación importante consiste en optimizar los modelos inspirados en redes neuronales que simulan las capas neuronales del cerebro humano. En [Jimenez-Fernandez-2012], [Perez-Peña-2013], [Serrano-Gotarredona-2009] se propone un controlador PID basado en redes neuronales de impulsos embebido en FPGAs. La codificación basada en impulsos imita el funcionamiento de la neurona. Las neuronas de

impulso son excitadas por corrientes de pulsos (impulsos), y su salida es solo otra corriente de impulsos. En las referencias anteriores, esta información se emplea para procesar la información visual del proceso y realizar el seguimiento. En [Rostro-Gonzalez-2015] se propone un sistema de control del movimiento para un robot hexápodo basado en un CPG (del inglés *Central Pattern Generator*). El sistema CPG fue construido como una red neuronal de impulsos, que producen señales rítmicas para tres velocidades diferentes (caminar, trotar y correr) en el robot hexápodo. Todos estos trabajos con redes neuronales tratan de buscar la implementación de controladores bioinspirados para la toma de decisiones en tareas normalmente de navegación. Es habitual encontrar trabajos como [Baturone-2014], donde las redes neuronales deciden en función de conjuntos difusos. Estos controladores neuro-difusos se implementan fácilmente y con bajo coste en las FPGAs, como demuestra el gran número de artículos que tratan el tema de los controladores neuronales, difusos y neuro-difusos embebidos en FPGA. También aparecen trabajos donde se buscan técnicas basadas en redes neuronales capaces de evaluar otras redes neuronales. Adicionalmente a las propiedades inherentes de las FPGAs, como el paralelismo y la reconfigurabilidad, el enfoque descrito en [Bonabi-2014] hace que el sistema basado en FPGA sea un candidato adecuado para el estudio del control neuronal de robots y sistemas cognitivos. Para ello se describe un conjunto de técnicas para la implementación eficiente del modelo basado en Hodgkin-Huxley (H-H) de una red neuronal en la FPGA. Las arquitecturas de control emocional se están convirtiendo en soluciones prometedoras para el diseño de los robots del futuro. Las emociones motivan los comportamientos del robot para cumplir los objetivos diseñados. Sin embargo, la carga de trabajo de las emociones aumenta con la complejidad de las aplicaciones, lo que reduce la capacidad del procesador para llevar a cabo problemas más complejos. En [Dominguez-2014], [Dominguez-2017] se aprovecha el potencial paralelismo de las emociones para mejorar el rendimiento de los sistemas de control emocional, implementando los controladores en las FPGA Cyclone de Altera.

La reducción de costes de las FPGAs, su creciente capacidad y la posibilidad de mejorar el rendimiento de los controladores con tecnologías de hardware específicas han hecho que su uso en nuevos campos de aplicación relacionados con controladores se haya

incrementado. Debido a la gran cantidad de datos a procesar, los sistemas de visión por computador representan actualmente uno de los campos de investigación más importantes [Batlle-2002], [Díaz-2004], [Jimenez-Fernandez-2012], [Tu-2011], [Fiack-2015]. Estos sistemas de visión por computador permiten obtener información del entorno y muchas veces constituyen el principal sensor de los sistemas para controlar la posición o navegación de los robots en su entorno. Los sistemas de visión por computador pueden clasificarse en tres niveles principales [Adelson-1994]. El nivel inferior incluye la captura de imágenes y funciones simples basadas en píxeles, tales como operaciones aritméticas entre imágenes. El nivel intermedio incluye operaciones tales como segmentación, *matching*, preprocesamiento, convolución o estimación de movimiento. El nivel superior se aplica típicamente para reconocer, clasificar o interpretar la escena. En la actualidad, hay una gran cantidad de sistemas de visión basados en FPGA que emplean principalmente la computación paralela con el fin de aumentar la velocidad de procesamiento en estas aplicaciones de computación intensiva [Arias-Estrada-2001]. Las imágenes y vídeos de alta resolución requieren complejos algoritmos de compresión y codificación [Reaz-2005]. Estas aplicaciones requieren recursos sustanciales de procesamiento y es posible encontrar varios trabajos que emplean la computación paralela de FPGAs para ofrecer soluciones [Dubois-2008], [Lu-2013], [Reza-2004]. En estas aplicaciones, el procesamiento de imágenes basado en FPGA permite satisfacer requisitos tales como bajo consumo de energía, pequeña escala de circuitos y reconfigurabilidad de la arquitectura de hardware. En el Capítulo 4 se profundizará en los sistemas de visión artificial embebidos en FPGA, desarrollando más en detalle su estado del arte.

Como se ha descrito anteriormente, el tratamiento de imágenes puede aprovechar el procesamiento en paralelo de las FPGAs [Tu-2011]. La información visual obtenida del tratamiento de las imágenes proporciona información global sobre el espacio de trabajo y se integra progresivamente en los sistemas de control. En [Jimenez-Fernandez-2009], se propone un robot móvil inspirado en redes neuronales con un doble mecanismo de control basado en impulso para dos motores de corriente continua. Todos los problemas de procesamiento de imágenes también se llevan a cabo en una FPGA (captura, procesamiento y seguimiento de línea). Un enfoque similar se presenta en [Linares-

Barranco-2007], donde se emplea un dispositivo AER (del inglés *Address-Event Representation*) para la detección visual, el procesamiento y finalmente el accionamiento de un robot. En un dispositivo de este tipo, los eventos se codifican en una dirección, se envían a través del bus y se decodifican. Los robots móviles de cuatro patas son un sistema eficaz para monitorizar una vivienda en ausencia de los dueños. El procesamiento de las imágenes para poder realizar la navegación autónoma de estos robots requiere normalmente gran capacidad de proceso. En [Peng-2016] se implementa el procesamiento de imagen para un robot de este tipo en una FPGA. En [Lentaris-2012], se presenta un diseño de hardware/software e implementación para la localización de robots en las misiones de *Mars Rover*. Este último documento propone una arquitectura de sistema implementada en una FPGA Virtex-6 de Xilinx para procesar las imágenes obtenidas, realizar el SLAM (del inglés *Simultaneous Localization And Mapping*) visual, la reconstrucción del mapa en 3D y obtener la ubicación del *rover* en el mapa. Ciertos trabajos tratan de resolver el problema de tener que procesar todo el fotograma en busca de la información necesaria para la navegación autónoma [Fiack-2015]. Este último trabajo describe un sistema de control en tiempo real embebido en una FPGA, que se basa en la atención visual para aprender acciones específicas en cada lugar reconocido por el robot. El modelo atencional permite reducir el procesamiento del sistema de visión artificial a unas pocas regiones del campo visual. En [Lorenz-2013], se presenta un sistema automático de alta precisión para la medición del nivel de líquido en aplicaciones de destilación de membrana. Este enfoque se basa en el principio de triangulación láser con dos láseres y una cámara. La medición del nivel se obtiene mediante una FPGA que realiza el procesamiento de la imagen. En [Marín-2007], [Marín-2009] se define el protocolo SNRP (del inglés *Simple Network Robot Protocol*), que permite la integración de robots de red y sensores. En este caso, se ha utilizado una FPGA para implementar un sistema de visión en tiempo real que proporciona servicios SNRP a la red. Utilizando el módulo de visión por computador embebido en la FPGA y el protocolo SNRP es posible implementar algoritmos de control visual sobre robots industriales. Es evidente que los algoritmos de procesamiento de imagen se benefician de las posibilidades que ofrece una FPGA para implementar procesos paralelos. Sin embargo, no hay que olvidar otro aspecto

importante en el proceso de captura y procesamiento de imágenes. En [Geier-2016] se muestra un trabajo en el que se presta atención a la forma en que los datos son transmitidos desde la cámara hacia la FPGA para poder ser procesados. Se describe un protocolo que aprovecha las características de nuevo de las FPGAs para utilizar el gran ancho de banda de los dispositivos GigE (Gigabit Ethernet). Para ello, se reencaminan todas las tramas Ethernet entrantes utilizando SG/DMA Proxying, donde se separan los datos de imagen de aquellos que no contienen información de imagen sin uso alguno de la CPU ni de la intervención de memoria, realizando así la adquisición de imágenes a la tasa máxima que proporciona la línea completa de Gigabit Ethernet.

Las FPGAs han sido utilizadas con éxito en diferentes sistemas como el control de velocidad de motores asíncronos de imanes permanentes [Nguyen-2014], emulación de máquinas eléctricas [Roshandel-2015], implementación de métodos de control para generación de pulsos y balanceo de voltaje [Wei-2015], etc. Dentro del campo del control de robots, es interesante mencionar trabajos previos como [Vachhani-2009], el cuál presenta soluciones hardware eficientes para dos tareas robóticas de alta movilidad. De esta manera, es posible encontrar arquitecturas para el control de robots que integran FPGAs en algunos componentes de su diseño [Shao-2006], [Shao-2007]. En este último caso, los algoritmos se dividen en una parte lineal, que se implementa en un FPGA, y una porción no lineal, que se implementa en un DSP. Previo al control de un robot complejo de muchos grados de libertad, los primeros trabajos en los que se utiliza una FPGA tratan de resolver el control de un solo eje lineal [Jeon-2002], [Osornio-Rios-2007], [Osornio-Rios-2009] con el fin de hacer que los robots industriales y las herramientas de control numérico por ordenador (CNC) realicen tareas de manera eficiente. También es habitual encontrar controladores de motores paso a paso embebidos en FPGA. En [Jabeen-2015] se describe una metodología de verificación formal para 6 tipos de controladores de motor paso a paso embebido en FPGA. Dentro del propio control de una estructura robótica más compleja, uno de los controladores básicos más utilizados es el control cinemático inverso. El cálculo de la cinemática inversa en robots manipuladores consume mucho tiempo de CPU y ciertamente ralentiza la velocidad de movimiento de los robots. Para resolver este problema, en [Kung-2014] se propone una implementación hardware de control de

cinemática inversa para robots manipuladores embebidos en FPGA. Para probar el controlador, se realiza el control de dos robots sencillos: un robot manipulador de tipo articulado y un manipulador de tipo SCARA. Si se tratan aplicaciones robóticas más complejas, por ejemplo, en [Chang-2006] se presenta un sistema de control basado en FPGA para lograr una arquitectura de hardware ideal para el control de impedancia de una mano robótica. En la última década, muchos centros de investigación han centrado sus esfuerzos en obtener manos robóticas lo más parecidas a las manos humanas [LiuH-2007], [LiuH-2008]. En este trabajo de Liu et al. diseñan una mano robótica utilizando una FPGA para obtener información de cada uno de los cuatro dedos. También se usa otra FPGA para la palma de la mano. Todo el intercambio de información se realiza gracias a la velocidad de envío de datos de las FPGAs. Siguiendo con el trabajo de implementar controladores para manos robóticas sobre FPGAs, en [ZhangT-2016] se propone un controlador de impedancia adaptativo para regular y realizar seguimientos de trayectorias de fuerza con compensación de fricción para una mano robótica de cinco dedos. Los resultados experimentales demuestran que se puede lograr un seguimiento preciso de la fuerza y una respuesta de fuerza y pares estable en entornos inciertos de rigidez y posición desconocidas. El control de impedancia permite a los robots manipuladores interactuar con su entorno. Es por ello por lo que es un tipo de control muy empleado para manos robóticas o robots manipuladores. Las FPGAs permiten en estos casos aumentar el ancho de banda del control de fuerza y minimizar la ondulación del par articular en robots con articulación flexible [HuangJB-2008]. Las soluciones basadas en FPGA han ganado relevancia en aplicaciones industriales para el control de posición y velocidad de actuadores eléctricos. Sin embargo, los controladores basados en FPGA tienen un amplio espectro de aplicaciones tales como el desarrollo de controladores para manipuladores robóticos con actuadores neumáticos. En [Juan-Manuel-2016] se presenta un trabajo en el que se implementa un control PD integrado en una FPGA para mover un robot manipulador neumático de un grado de libertad. El trabajo presentado en [ZhangL-2012] describe una arquitectura de control basada en FPGA que puede aplicarse fácilmente a cualquier robot. En [LeeWK-2006] se describe un controlador basado en FPGA para un brazo robot humanoide que implementa varias tareas como controladores

PID, comunicación, contadores de encóder o generadores PWM (del inglés Pulse Width Modulation). En [Wen-Hong-2013] se propone aplicar un controlador integrado basado en FPGA para robots modulares. Basado en una FPGA Cyclone IV de Altera, en [Kho-2016] se diseña e implementa un pequeño robot autónomo, donde se incluye desde la lectura de sensores para el posicionamiento, hasta el controlador que permite la navegación autónoma del robot. También se encuentran recientemente trabajos que aprovechan las propiedades de las FPGAs para realizar el control de robots móviles (con ruedas) [HuangHC-2015], [HuangHC-2016c]. Para el control del robot móvil descrito en estos últimos artículos, se utiliza un sistema evolutivo difuso con optimización de colonia de hormigas (ACO).

El uso de FPGAs permite utilizar una tecnología específica de hardware reprogramable para la implementación de controladores visuales. Los sistemas de control visual clásicos realizan el movimiento punto a punto de un robot usando información visual [Chaumette-2006]. En contraste con los controladores visuales clásicos, el enfoque dinámico o directo de control visual [Kelly-2000] realiza el control por par articular directamente mediante el uso de información visual. Por lo tanto, la ley de control visual directo proporciona los pares que se aplicarán a las articulaciones del robot para realizar su guiado. Como se describe a lo largo de esta Tesis, la implementación de un sistema de control visual directo embebido totalmente en una FPGA permite mejorar estos sistemas, reduciendo los retardos de procesamiento y obteniendo además retardos estables en la retroalimentación. Además, la posibilidad de realizar reconfiguraciones parciales permite modificar secciones de la lógica implementada en la FPGA, ajustando según la necesidad la funcionalidad y dinámica de los controladores a las propiedades de la aplicación a desarrollar. Por lo tanto, el uso de una FPGA permite implementar una arquitectura paralela dedicada que se puede adaptar a las necesidades del sistema en tiempo de ejecución.

Dentro del ámbito de los sistemas de control visual, sólo hay algunas implementaciones que integran FPGAs en algún elemento del sistema de control visual. La mayoría de los retardos en estos sistemas se deben a tareas de procesamiento de

imágenes. Esta es la razón por la cual hay algunos trabajos como [Liyanage-2014], que optimizan el proceso de captura de imágenes a través de la implementación de hardware que permite una FPGA. También en [WangS-2016] se emplea una FPGA para permitir procesamiento de imagen en tiempo real para posteriormente realizar un posicionamiento, esta vez mediante control visual basado en posición. Sin embargo, en este último trabajo el control no se realiza en la FPGA. Otra aproximación en la que se utiliza la FPGA para hacer posible un procesamiento de imagen más eficiente y en tiempo real es [Dhipa-2015]. Al paralelizar el proceso, se mejora la velocidad de procesamiento de una imagen estereoscópica. Los datos y la tarea se paralelizan mediante el uso de cauces segmentados en una FPGA. Las imágenes de los pares estéreo proporcionan información 3D para realizar posteriormente tareas de posicionamiento mediante control visual basado en posición. Adicionalmente, es posible encontrar sistemas de control visual indirectos embebidos que integran tanto el procesamiento de imágenes como el control [HuJS-2011], [Jwu-Sheng-2011]. En [Tu-2011], una FPGA se ha utilizado con éxito para lograr un control robusto de un péndulo invertido utilizando un coprocesador de imágenes basado en FPGA y [ParkJH-2009] presenta un sistema de control visual que consiste en un sistema de visión por computador y un robot para mover la placa plana sostenida por el robot para que la bola en la placa pueda seguir una determinada trayectoria deseada. Los trabajos iniciados en [ZhangL-2012] se amplían a sistemas de control visual en [ZhangL-2014]. En este último trabajo se propone una arquitectura abierta para el control visual de robots. La arquitectura y los núcleos IP relacionados se desarrollan a través de un modelo de diseño basado en componentes que interpreta un sistema robótico por componentes de computación, comunicación, configuración y coordinación, junto con la composición de estos componentes y cajas negras. El núcleo IP del controlador del motor se diseña como hardware abierto. El software abierto implica el uso de OROCOS [Bruyninckx-2001]. El rendimiento del control visual se evalúa mediante el acoplamiento visual de un robot Performer MK2 y un robot LWR de KUKA. En [PérezC-2008] se describe el modelado e identificación de parámetros de un robot cartesiano de 3 gdl. La implementación de un esquema de control visual indirecto y la adquisición/procesamiento de la imagen se realiza en una FPGA con objeto de acelerar el

tiempo de cálculo. Sin embargo, no es posible encontrar arquitecturas basadas en FPGA que implementen controladores visuales directos para guiar robots manipuladores utilizando información visual. Ésta es una de las principales aportaciones de la arquitectura propuesta tal y como se describirá en el Capítulo 5 de esta Tesis.

2.3 Control Visual

En este apartado se describen los sistemas de control visual con el objetivo de especificar su notación, funcionamiento y enmarcar las aportaciones realizadas en esta Tesis, y que se detallan en los siguientes Capítulos. Para ello, se comienza describiendo qué son los sistemas de control visual, se describen sus tipos básicos atendiendo a tres clasificaciones definidas por tres parámetros concretos: la posición de la cámara respecto al robot; el tipo de entrada y consigna del controlador; y la salida del controlador, es decir, qué órdenes de movimiento genera el controlador hacia el robot.

2.3.1 Definición

Los sistemas de control visual (o su correspondiente en inglés *visual servoing*) permiten el posicionamiento de un robot respecto a un objeto del espacio de trabajo mediante el uso de información visual en la realimentación del sistema. Uno de los primeros trabajos en los que se emplea el término *visual servoing* fue en [Hill-1979], donde se plantea la utilización del control en bucle cerrado introduciendo información visual en la realimentación. Ya en 1980, Sanderson y Weiss establecen una clasificación de los sistemas de control visual en basados en posición y en imagen, describiendo asimismo el primer sistema de control visual basado en imagen [Sanderson-1980]. Ésta es la principal clasificación de los sistemas de control visual, y se verá con más detalle en el apartado 2.3.2. Desde la aparición de estos primeros sistemas de control visual a principios de los 80, su evolución ha sido lenta, debido principalmente a las capacidades de procesamiento de los ordenadores y sistemas de visión artificial de la época, que impedían procesar una escena a una frecuencia suficiente. Una de las primeras líneas de investigación abiertas dentro de este campo es la desarrollada por [Chaumette-1990], consistente en la aplicación de la función de la tarea [Samson-1991] en el caso de utilizar la información captada por

una cámara. A partir de entonces han surgido un gran número de autores interesados en distintos aspectos del control visual. En las siguientes referencias es posible encontrar una revisión de las principales líneas de investigación dentro de los sistemas de control visual [Chaumette-2006], [Chaumette-2007], [Chesi-2010], [Janabi-Sharifi-2011], [Tao-2016]. En el Apartado 2.3.2 se citarán algunas de las líneas de investigación más activas y recientes dentro del campo de los sistemas de control visual, prestando una mayor atención a aquellas relacionadas con esta Tesis.

Los principales componentes que conforman un sistema de control visual son los que se representan en la Figura 2.1 y que a continuación se detallan:

- *Referencia.* Se trata de la configuración final que se desea que alcance el robot. El tipo de referencia utilizada depende de si se emplea un controlador basado en posición (la referencia es una posición 3D a alcanzar) o si se emplea un controlador basado en imagen (la referencia es un conjunto de características visuales deseadas).
- *Controlador*. Es el sistema encargado de realizar el guiado del robot haciendo uso de la información visual hasta conseguir alcanzar la referencia. El tipo de controlador dependerá, entre otras cosas, de si se realiza un control basado en posición, en imagen o si el control es directo o indirecto. Las principales características de estos controladores se indicarán en el Apartado 2.3.2.
- *Sistema de Visión Artificial.* Este bloque representa la realimentación del sistema de control visual y se encarga de realizar la extracción de la información visual requerida para guiar al robot. En el caso de un controlador basado en imagen, este componente simplemente extrae información visual. Sin embargo, cuando se emplea control basado en posición, este bloque debe determinar la posición 3D del objeto implicado en la tarea a partir de la información visual capturada.

Para terminar con el estudio de los diferentes componentes que conforman los sistemas de control visual, es necesario profundizar en uno de los elementos más importantes, como es la ubicación del sistema de visión. La información visual puede ser adquirida por

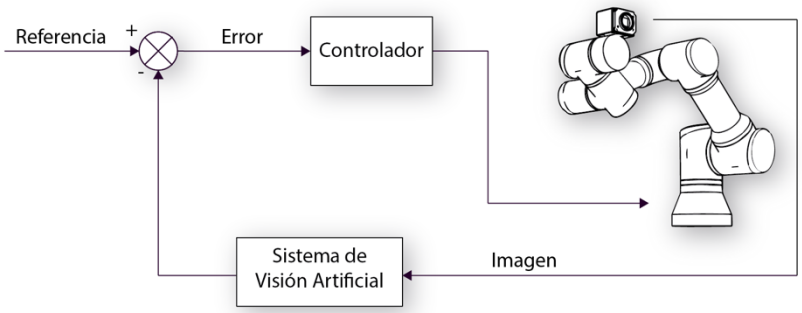


Figura 2.1 Esquema genérico de control visual

una cámara montada directamente sobre el robot manipulador, en cuyo caso el movimiento del robot induce movimiento en la cámara (es la configuración que se puede ver en la Figura 2.1), o la cámara puede ser fijada en el espacio de trabajo de manera que se puede observar el movimiento del robot a partir de una configuración estacionaria. A la primera configuración (representada en la Figura 2.2) se le llama configuración en el extremo del robot (del inglés *eye-in-hand*) y a la segunda configuración (representada en la Figura 2.3) se le llama configuración externa al robot (del inglés *eye-to-hand*). Cuando se emplea una cámara en el extremo del robot, los objetos del espacio de trabajo se encuentran dentro de su campo visual y se tiene una relación constante entre el sistema de

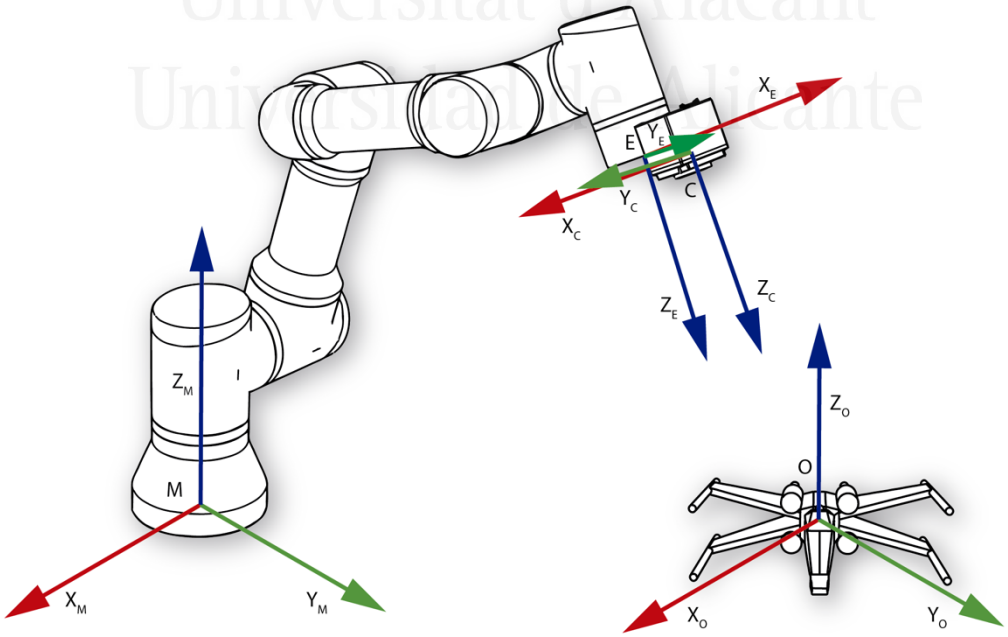


Figura 2.2 Cámara en el extremo del robot

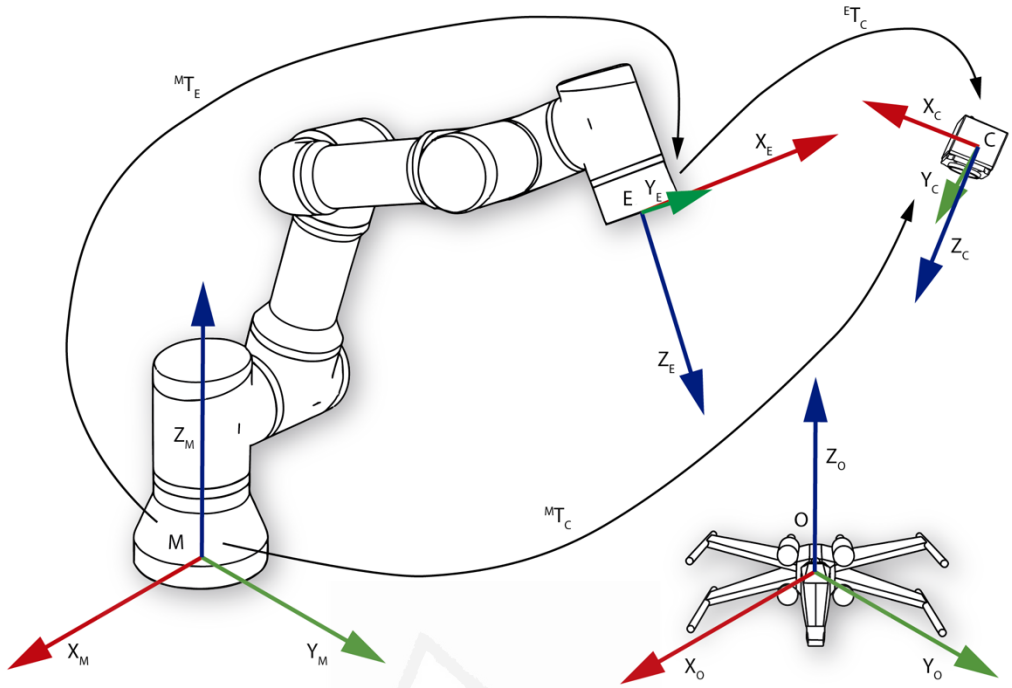


Figura 2.3 Cámara externa al robot

coordenadas de la cámara y el del extremo del robot, ETC. Cuando se utiliza una cámara externa al robot la relación de posiciones entre la cámara y el sistema de coordenadas situado en la base del robot, MTC, es fija. Por el contrario, no existe una relación constante entre las posiciones de la cámara y el extremo robot, ETC. A lo largo de la Tesis se empleará una configuración *eye-in-hand*, de forma que el objeto del cual se extraen las características se encuentra en el campo de visión de la cámara. Además, como ya se ha indicado y puede verse en la Figura 2.2, la relación entre los sistemas de coordenadas de la cámara y del extremo del robot se mantendrá constante a lo largo de la tarea.

Como se ha indicado anteriormente, el objetivo de un sistema de control visual será el de realizar el posicionamiento de un robot empleando visión artificial. En [Pérez-2015b] se presentó un trabajo sobre Matlab que permite la simulación de este tipo de sistemas. En el siguiente apartado se realizará una nueva clasificación de los sistemas de control visual, con el objetivo de ir centrando progresivamente el tipo de sistema de control visual implementado, así como las aportaciones realizadas.

2.3.2 Tipos de sistemas de control visual

Este apartado tiene como objetivo realizar una revisión sobre los diversos tipos de sistemas de control visual. En primer lugar, se clasificarán los sistemas de control visual en basados en posición y basados en imagen. Los basados en posición son aquéllos en los que la referencia del sistema de control es una localización tridimensional deseada. Sin embargo, en los basados en imagen la referencia viene dada directamente en el espacio de la imagen. Esta última será la aproximación empleada en esta Tesis. Esta clasificación inicial se realizará considerando una implementación "clásica" del controlador. Estos controladores clásicos se corresponden también con lo que suele denominarse como controladores indirectos. En este caso, la acción de control viene dada como velocidades a aplicar al robot y se supone la dinámica del mismo como despreciable. Cuando se aplica un controlador indirecto se tiene en cuenta únicamente aspectos cinemáticos del robot. Sin embargo, en los controladores directos la acción de control viene dada habitualmente como pares a aplicar a las articulaciones. Para cada uno de los tipos de control visual descritos a lo largo de este apartado, se indicarán las principales investigaciones relacionadas con la presente Tesis Doctoral, haciendo énfasis en las aportaciones realizadas.

2.3.2.1 Control visual indirecto basado en posición

En los sistemas de control basados en posición se dispone de un modelo del espacio de trabajo de forma que, a partir de las características extraídas del sistema de visión, se estima la localización del objeto a alcanzar respecto al sistema de coordenadas de la cámara. Esta localización estimada se compara a su vez con la deseada o de referencia; la diferencia entre ambas localizaciones será la entrada al regulador. Habitualmente, para reconstruir la posición y orientación 3D del objeto implicado en la tarea ($\boldsymbol{p}^C$, $\boldsymbol{\varphi}^C$), no sólo se ha de disponer de un modelo del espacio de trabajo, sino también de un modelo calibrado de la cámara empleada. En este tipo de control visual, la referencia del sistema será la posición y orientación deseada del robot o de un objeto existente en el espacio de trabajo.

En la Figura 2.4 se representa el esquema básico de un controlador basado en posición. Este esquema, que se corresponde con un controlador indirecto, presenta un primer nivel de realimentación proporcionada por los controladores de las articulaciones, mientras que el segundo nivel corresponde a una realimentación visual. En el esquema se indica la necesidad de reconstruir la localización 3D del objeto implicado en la tarea.

Dentro del ámbito de los sistemas de control visual basados en posición cabe citar desde algunos trabajos clásicos como los de [Allen-1991] hasta otros más actuales como los de [ZhaoYM-2017]. Entre ambos trabajos cabe citar algunos destacables como los de [Wilson-1996], que describen la utilización de visión para el posicionamiento de un robot respecto a un objeto del cual se realiza la reconstrucción de su posición. Como se ha indicado anteriormente, en los sistemas de control visual basados en posición es necesario estimar la localización tridimensional del objeto implicado en la tarea a partir de información visual. En [HuX-2013] se trata el problema de la calibración de la matriz de transformación de cámara a extremo del robot, que puede afectar especialmente a la precisión del control visual basado en posición. Dentro de los trabajos de reconstrucción a partir de primitivas de tipo punto, cabe resaltar la contribución de [DeMenthon-1995], consistente en una técnica iterativa que permite la estimación rápida de la posición de un objeto a partir de 4 puntos o más no coplanares, o a partir de 4 puntos coplanares [Oberkampf-1996]. Otros autores [David-2004] desarrollan una modificación de este algoritmo para permitir simultáneamente la correspondencia y determinación de la posición a partir de los puntos. El uso de múltiples cámaras permite reconstruir por consenso la posición de un objeto visto desde las distintas cámaras [Jorstad-2008].

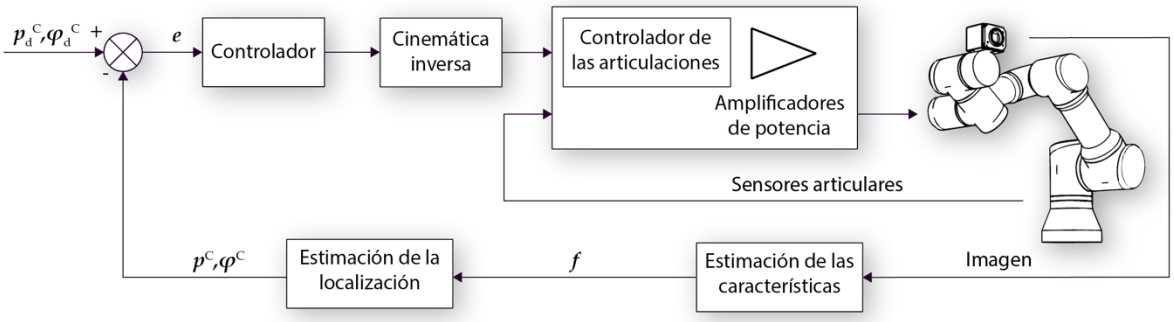


Figura 2.4 Controlador indirecto basado en posición

Asimismo, es posible encontrar trabajos encaminados a estimar la localización utilizando primitivas como segmentos. En esta línea, destacan los estudios de [Horaud-1997] consistentes en una adaptación del algoritmo de [DeMenthon-1995] para el caso de extracción de segmentos. En [Miljković-2013] se utiliza control visual basado en posición para el guiado inicial de un robot autónomo capaz de mover material en una fábrica. La estimación del estado de la posición y orientación del robot se determina en términos de SLAM monocular, a través de un filtro extendido de Kalman alimentado con una red neuronal, definido como filtro Neuronal Extendido de Kalman (NEKF). Este filtro NEKF se utiliza para modelar perturbaciones desconocidas y para mejorar el modelo de estimación del estado del robot. También en [Parsapour-2013] y en [Larouche-2015] se utilizan estimadores derivados del filtro de Kalman para estimar la posición y orientación de la cámara respecto a un objeto móvil.

De la misma manera, cabe resaltar las investigaciones de [Cervera-2001], [Cervera-2003], [Prats-2008], sobre sistemas estereoscópicos para su utilización en sistemas de control visual basados en posición, así como la inclusión de características 3-D en los sistemas de control visual basados en imagen. Aunque es bien conocido que este tipo de sistemas de control visual permite obtener una trayectoria que se aproxima a la lineal entre la posición inicial y la deseada [Deng-2003], aún en la actualidad se siguen estudiando problemas de convergencia y robustez derivados [Wang-2012], [ParkDH-2012], [LeeJ-2013]. Los estudios de [Thuilot-2002] tratan el problema de la pérdida del objeto durante la tarea de control visual basado en posición, para lo cual generan de manera iterativa una trayectoria que mantenga al objeto siempre en el campo de visión de la cámara. Los trabajos de [Lippiello-2003], [Lippiello-2007] se centran en el desarrollo de sistemas de control visual basados en posición haciendo uso de sistemas multi-cámaras para el seguimiento de objetos en movimiento. En [Ibarguren-2014], [Abdul-Hafez-2014] se propone un controlador visual basado en posición que usa un filtro de partículas para tratar de minimizar las perturbaciones producidas por las diferentes fuentes de ruido de los entornos industriales. En la actualidad los controladores basados en posición se vienen aplicando no sólo al guiado de robots manipuladores sino también a otros entornos como el control de los diferentes grados de libertad de vehículos aéreos [Mahony-2012], [Mohta-

2014], [Lippiello-2016]. En este último trabajo de Lippiello et al. se utiliza, al igual que en [Miljković-2013], un control híbrido, de forma que el control visual basado en posición se emplea para posicionamientos menos precisos, y el control visual basado en imagen cuando se requiere precisión (normalmente porque se está ya cerca de la posición objetivo). Sin embargo, la necesidad anteriormente citada de reconstruir posiciones 3D, con la problemática que implica, junto con la robustez de los sistemas de control basados en imagen han hecho que estos últimos se vengan aplicando en mayor medida en la actualidad [Martinet-2015]. Como se ha indicado anteriormente, ésta no ha sido la aproximación empleada en la Tesis ya que se ha optado por la utilización de una arquitectura basada en imagen que no requiera reconstruir la localización tridimensional del objeto implicado en la tarea.

2.3.2.2 Control visual indirecto basado en imagen

Cuando se emplea control visual basado en imagen, el regulador tiene como finalidad generar acciones de control de forma que las características visuales de las imágenes obtenidas vayan convergiendo progresivamente a las deseadas. En la Figura 2.5 se muestra el esquema de un controlador indirecto basado en imagen. Como se observa en esta última figura, en la realimentación del bucle de control se realiza un proceso de extracción de características visuales que en este caso se corresponde con cuatro puntos en la imagen. A diferencia de los controladores basados en posición, no es necesario calcular la posición 3D del objeto implicado en la tarea. De esta manera, el control se realiza directamente en el espacio imagen.

Aunque la literatura en la que se hace uso de controladores visuales indirectos basados en imagen es muy extensa, en los trabajos de [Chaumette-2006] y [Chaumette-2007] se realiza una revisión de los principales trabajos y problemáticas relacionadas con los controladores indirectos basados en imagen.

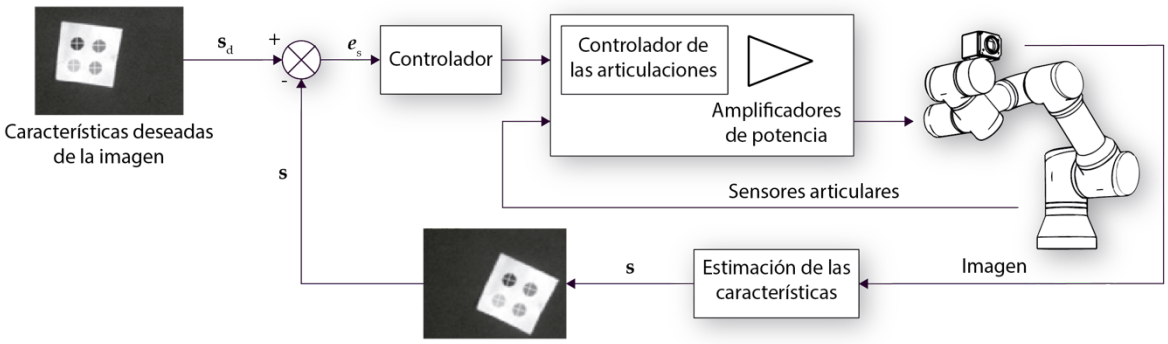


Figura 2.5 Controlador indirecto basado en imagen

Uno de los elementos que más influyen en el comportamiento de los controladores basados en imagen es la denominada matriz de interacción. Esta matriz relaciona la velocidad de movimiento del objeto observado en el espacio imagen con la velocidad correspondiente en el espacio 3D. Como se indicará posteriormente, esta matriz es empleada en el controlador y depende de distintos parámetros. A lo largo de los últimos años se ha trabajado con el objetivo de realizar su estimación, o demostrar la robustez del sistema de control visual ante errores de dichos parámetros [Musić-2014]. Uno de los parámetros de los que depende es la distancia entre la cámara y el objeto observado. Actualmente es posible encontrar distintos trabajos [De Luca-2008], [LeeD-2012] que estiman esta profundidad sin conocimiento previo, o que emplean una aproximación adaptativa para obtener un correcto comportamiento ante incertidumbres en la profundidad [Cheah-2012], [Zhang-2015], lo que dota de mayor robustez a este tipo de controladores. Una importante línea de investigación es la abierta por [Malis-2003] con los sistemas de control visual no calibrados e invariantes a errores en los parámetros intrínsecos de la cámara, presentando, por ejemplo, un correcto comportamiento ante errores en la determinación del valor de la focal de la cámara utilizada. El problema del uso de la homografía es el cálculo de la distancia entre la cámara y el plano de referencia. Para calcular esta constante, en [ChenJ-2005] se proponen el uso de técnicas adaptativas. Otros trabajos, como los descritos en [GarciaGJ-2004] y en [XieWF-2009] obtienen la profundidad en línea a partir de los valores de medida de un sensor láser de distancias. También es habitual la estimación de esta matriz utilizando redes neuronales [Kosmopoulos-2011], [Nadi-2014]. En la actualidad, la mayoría de las implementaciones

utilizan una simplificación de esta matriz, consistente en determinar su valor en el equilibrio (configuración deseada), para evitar el cálculo de la profundidad en línea. Esta será la aproximación que se empleará en la Tesis.

Relacionados con los trabajos anteriores se encuentran los estudios realizados por distintos autores [MaZ-2015], [Nematollahi-2009], [Chaumette-1998], relativos a los problemas de singularidad, robustez y estabilidad en los sistemas de control visual basados en imagen. Este último autor desarrolla un estudio acerca de la estabilidad y convergencia de estos sistemas indicando distintas situaciones que pueden provocar mínimos locales. Con posterioridad, en [Chaumette-2006] se hace una revisión de estos problemas de estabilidad. Para evitar estos problemas de singularidad y mínimos locales se desarrolló un trabajo con anterioridad [Marchand-1996] en el que se proponía la inclusión de tareas secundarias dentro de la propia función de tarea para garantizar la convergencia. En 2013, se propone el uso de características visuales invariantes para dotar a los sistemas de control visual basado en imagen de mayor robustez [Tahri-2013]. Más recientemente, en [Briot-2016] se utiliza el concepto de robot oculto para determinar casos de singularidad en el caso de uso de puntos como características visuales del controlador. Con el objetivo de evitar los problemas de convergencia citados, cabe reseñar asimismo los trabajos de [Corke-2001] relativos a control visual particionado. En estos trabajos se propone una estrategia de control que trata de forma separada distintos grados de libertad de la tarea para garantizar la convergencia ante distintas situaciones como se indicaba en [XieWF-2009]. En [Mansard-2009] se define el concepto de controlador basado en la función de tarea para tener en cuenta restricciones unilaterales como los límites de las articulaciones del manipulador. Estos problemas de límites articulares se tratan en [Siradjuddin-2012] a partir de una estimación iterativa de la matriz de interacción para un robot redundante. El empleo de robots redundantes produce también problemas de estabilidad. En este caso, no sólo es necesario realizar de forma correcta el guiado del extremo del robot, sino también es necesario resolver la redundancia estableciendo la configuración articular más adecuada en cada caso [Siradjuddin-2014]. Este es el caso del trabajo presentado por [GarciaN-2011] en el que se emplearon dos robots redundantes de

forma simultánea. En [Fang-2012] se utilizó control visual 2.5D para el guiado de robots redundantes realizando un estudio de estabilidad basado en Lyapunov.

En algunas investigaciones, con el objetivo de mejorar el comportamiento del robot, se propone combinar la información visual con otros sensores. Por ejemplo, los sistemas de control visual/fuerza híbridos proponen el control simultáneo de direcciones distintas. Una dirección es controlada empleando información visual y otra distinta usando control de fuerza [GarciaGJ-2009a], [Siciliano-2010]. En la Tesis se emplearán robots que podrán ser redundantes respecto a la tarea descrita en el espacio imagen. El filtro de Kalman [Kalman-1960] es una de las técnicas más extendidas para implementar filtros Bayesianos [Simon, 2001]. El filtro de Kalman estándar aproxima las estimaciones de un filtro Bayesiano empleando distribuciones gaussianas, y representa el modelo de transición del estado y el modelo de observación como relaciones lineales estocásticas del estado. Dentro del ámbito de los sistemas de control visual es posible encontrar aproximaciones previas que emplean filtros de Kalman para el tracking de personas. [Foxlin-2005] emplea un filtro de Kalman en un sistema de navegación para el seguimiento de peatones usando sensores inerciales y un receptor GPS. Este tipo de filtro de Kalman es propuesto también por [Roetenberg-2007], empleando las diferencias entre un sistema de tracking inercial y un tracker magnético. En [Caron-2006] extienden la definición de filtro de Kalman para incluir múltiples sensores donde cada sensor es ponderado de acuerdo a variables difusas. El trabajo propuesto por [Ribo-2004] utiliza el filtro de Kalman extendido en un sistema de seguimiento compuesto por una cámara y un sistema inercial. Sobre la base del filtro de Kalman extendido, en [Janabi-Sharifi-2010] se estima la posición del objeto utilizando un filtro adaptativo iterativo extendido de Kalman. La estimación de objetos en movimiento para su realimentación en la ley de control es quizás la aplicación más empleada de estos estimadores basados en el filtro de Kalman [LiuC-2012]. Más actualmente, en [Van-2016] se utiliza el filtro de Kalman para la estimación de fallos en el movimiento de la cámara o en la adquisición de una imagen para un sistema de control visual indirecto basado en imagen.

Un problema que ha sido objeto de estudio por distintos investigadores dentro del campo de los sistemas de control visual basados en imagen es la necesidad de mantener las características visuales extraídas dentro del campo de visión de la cámara durante toda la tarea de control visual [Mezouar-2002], [Comport-2003], [Chesi-2005], [Schramm-2006], [WangY-2009], [Shademan-2012], [Doignon-2015], [Cazy-2015]. Una posibilidad de evitar la pérdida de características en la medida de lo posible es utilizar cámaras panorámicas, es el caso de trabajos como [Hadj-Abdelkader-2008], [Mariottini-2008], [Cervera-2008], [Tahri-2009], [Tahri-2012], [Aliakbarpour-2014]. Existen otras aproximaciones [Mezouar-2003], [Schramm-2006], [GarciaGJ-2009b], [Copot-2012], [Kazemi-2013], [WangK-2014], [Keshmiri-2017] que emplean seguimiento de trayectorias en vez de un conjunto de características visuales deseadas fijas. En la Tesis se hará uso de esta última aproximación ya que el problema de control visual es formulado como el seguimiento de una trayectoria dependiente del tiempo.

2.3.2.3 Control visual directo basado en posición

En este apartado se indican las principales características de los sistemas de control visual directo basado en posición. Como ya se ha especificado, en los sistemas de control visual indirectos, el controlador calcula la velocidad que se debe aplicar al extremo del robot, o la velocidad articular, para conseguir posicionar la cámara respecto a un objeto de referencia. Para ello, se hace uso del controlador interno del robot, que traduce esas velocidades a pares y fuerzas sobre las articulaciones. Sin embargo, empleando un esquema de control directo como el que se muestra en la Figura 2.6, se elimina el bucle interno de realimentación de los servomotores de forma que la información visual es empleada directamente para generar las corrientes o pares articulares a aplicar a cada uno de los motores del robot. Estos controladores integran no sólo la cinemática del robot, sino que también tienen en cuenta el comportamiento dinámico. El resultado es un controlador más rápido que consigue reaccionar mejor ante cambios en la referencia o ante movimientos del objeto observado. Tal y como se representa en la Figura 2.6, en el caso de control visual directo basado en posición la referencia es una localización tridimensional deseada.

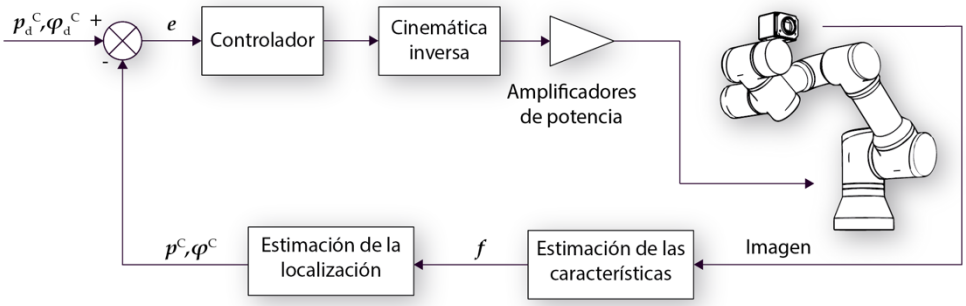


Figura 2.6 Controlador directo basado en posición

No es posible encontrar muchos trabajos en la literatura sobre controladores visuales basados en posición que tengan en cuenta la dinámica de los robots manipuladores y no sólo la cinemática. A los problemas de calibración de la cámara o de la relación entre la cámara y el extremo del robot para el cálculo de la posición y orientación en función de la información visual, se añade en estos sistemas la incertidumbre en la estimación de los parámetros dinámicos del robot. Sin embargo, estos sistemas son mucho más rápidos y precisos que los sistemas de control visual indirecto, como se explica en [Dahmouche-2012]. En [Bishop-1999] se desarrolla un esquema de control por dinámica inversa en el que se debe disponer de un conocimiento preciso de la dinámica del robot. Los controladores por dinámica inversa alcanzan el objetivo de seguimiento de trayectoria mediante la linealización por realimentación de la dinámica no lineal del robot. El controlador propuesto por Bishop y Spong permite indicar una trayectoria para el extremo del robot, sin embargo, la trayectoria deseada no está codificada en la imagen, sino en el espacio Cartesiano 3-D, ya que se realiza un control visual basado en posición. Cid et al. [Cid-2008] describen una metodología para diseñar sistemas de control visual basado en posición para robots de 2 grados de libertad que trabajan sobre un plano con cámara fija. Los trabajos presentados por Vargas y Rubio [Vargas-2003] describen un sistema de control visual directo basado en posición donde se tienen en cuenta tanto el modelo geométrico del objeto como el modelo dinámico del robot. En esta línea, los trabajos realizados por Sebastián [Sebastian-2007], [Angel-2008] permiten realizar un control visual directo basado en posición sobre un robot paralelo desarrollado por su grupo de investigación, dado que se conoce con precisión su modelo dinámico. Es en el control visual de robots paralelos donde se encuentran más trabajos acerca del control

visual directo basado en posición. En [Palmieri-2012] se implementan dos controladores visuales directos, pudiendo así comparar el desempeño de los basados en posición con los basados en imagen. La conclusión a la que se llega en este último trabajo es que el comportamiento de los sistemas de control visual directos basados en imagen es sensiblemente mejor, debido sobre todo a la necesidad de los sistemas basados en posición de obtener una correcta calibración de la cámara. El control basado en imagen muestra una mayor sensibilidad al desplazamiento horizontal, como se observa en los experimentos realizados al mostrar componentes más altos del vector de velocidad en las direcciones X e Y.

2.3.2.4 Control visual directo basado en imagen

Por último, para finalizar la clasificación de los sistemas de control visual, en la Figura 2.7 se indican los principales componentes de un sistema de control visual directo basado en imagen. En esta configuración, la referencia está compuesta por un conjunto de características visuales deseadas que habrían de observarse una vez que la tarea haya finalizado. El error $\boldsymbol{e}_s$ especificado en el espacio imagen, es la entrada al controlador. Éste último se encarga de determinar los pares articulares necesarios (o corriente a aplicar a cada uno de los motores) para hacer converger las características extraídas hacia la deseadas. Debido a que dentro de esta temática se centrarán algunas de las aportaciones de la Tesis, en la sección 2.4 se describen las principales consideraciones para la implementación de los controladores visuales directos basados en imagen.

En la literatura es posible encontrar algunos investigadores que han trabajado en controladores visuales directos basados en imagen con el objetivo de obtener mejores tiempos de respuesta y mejor estabilización frente a los controladores indirectos. A continuación, se citan aquellas investigaciones más relacionadas con la Tesis Doctoral, haciendo énfasis en las aportaciones realizadas.

Como se menciona en [Kelly-1996], hasta mediados de la década de los 90 fueron pocos los controladores propuestos basados en visión que tenían en cuenta la dinámica no lineal

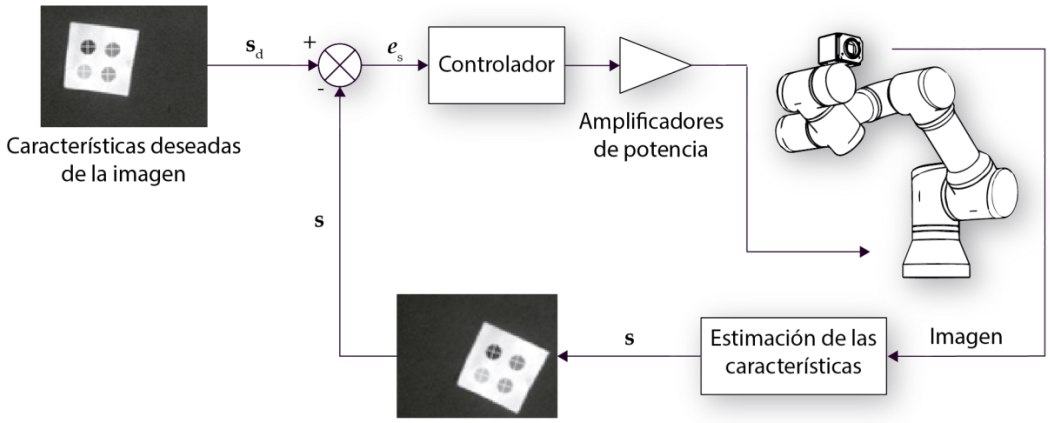


Figura 2.7 Controlador directo basado en imagen

de los robots manipuladores. En estas dos últimas décadas la tendencia sigue siendo la misma, aunque en los últimos años se aprecia un aumento de trabajos relacionados con el control visual directo basado en imagen para vehículos autónomos no tripulados. Sin embargo, centrando la atención de nuevo en los robots manipuladores, son pocos los trabajos donde se tiene en cuenta la dinámica del robot. Es decir, la mayoría de los controladores desarrollados se diseñan bajo la asunción de que el robot es un dispositivo de posicionamiento perfecto con una dinámica despreciable, lo que reduce el problema a un problema de control cinemático basado en las imágenes adquiridas por la cámara. El control visual dinámico es una técnica que permite obtener un comportamiento de alto nivel de los sistemas robóticos controlados por visión. La eliminación del lazo de control interno del robot permite un procesamiento de la imagen a una frecuencia muy elevada. Los trabajos existentes en la literatura en los que se describe el control visual dinámico de un robot tratan, en la mayoría de los casos, con robots con pocos grados de libertad. Uno de los primeros trabajos de control visual donde se tiene en cuenta la dinámica del robot es el realizado en [Miyazaki-1990]. El controlador propuesto por Miyazaki y Masutani se fundamenta en la aproximación de controlador basado en la Jacobiana traspuesta, técnica desarrollada por primera vez por Takegaki y Arimoto [Takegaki-1981]. El sistema de visión es modelado, sin embargo, como una simple transformación de rotación, no teniendo en cuenta la traslación. Kelly y Márquez [Kelly-1995] consideraron un modelo del sistema cámara-robot mucho más representativo que el propuesto en [Miyazaki-1990]. El controlador descrito sólo trata de resolver el problema del posicionamiento desde un

punto a otro conocido. Este método resuelve los problemas surgidos de la necesidad de obtener los parámetros intrínsecos de la cámara, sin embargo, requiere conocer de manera precisa la orientación de ésta. Este problema se resuelve en [Kelly-1996] donde se mejora el controlador propuesto en [Kelly-1995] para tener en cuenta la incertidumbre generada por la orientación de la cámara, permitiendo obtener una estabilidad asintótica local. El principal problema de los controladores desarrollados por Kelly et al. es que requieren un conocimiento exacto del término gravitacional del robot, y que la diferencia entre las orientaciones estimada y actual de la cámara se restringe al intervalo (-90^{o},90^{o}). Carelli et al. pretenden resolver, en [Carelli-1994], el problema de errores cometidos en la estimación de los parámetros dinámicos del robot. Para ello, emplean un controlador adaptativo que usa la información visual para compensar la incertidumbre de los parámetros dinámicos del robot. Más recientemente, este controlador basado en Jacobiana traspuesta se ha utilizado para controlar el posicionamiento y la orientación en el eje vertical de un cuadrirrotor [Ozawa-2012]. En este caso, se han utilizado los momentos de la imagen en lugar de los puntos característicos para la definición de las características visuales.

El controlador basado en la dinámica inversa, como el descrito en [Bishop-1999], utiliza la información visual directamente para controlar la posición relativa entre el robot y el objeto observado, es decir, es un control visual basado en imagen. Además, se demuestra la estabilidad asintótica local del método de control. Ya en la primera década del 2000, se sigue profundizando en la búsqueda de controles adaptativos que permitan resolver el problema de los errores cometidos en la estimación de los parámetros dinámicos. Así, Zergeroglu et al. diseñan un controlador adaptativo que tiene en cuenta la incertidumbre en la estimación de los parámetros del conjunto cámara-robot [Zergeroglu-2001]. En 2008, Wu y Li describen otro controlador adaptativo que, teniendo en cuenta los parámetros dinámicos del robot, trata de estimar en cada iteración los parámetros extrínsecos e intrínsecos de la cámara [Wu-2008]. Ya a principios de esta década, cabe reseñar los trabajos realizados por Cheah et al. [Cheah-2010] y [Cheah-2012] con el propósito de implementar un controlador visual directo adaptativo capaz de llevar a cabo la tarea correctamente ante errores en el modelo dinámico o algunos parámetros de la tarea, como

la profundidad o distancia entre la cámara y el objeto observado. En [WangH-2014] se propone una Jacobiana de la imagen que no depende de la distancia entre la cámara y el objeto. En general, para los controladores visuales directos, así como para los indirectos, sólo se ha demostrado su estabilidad localmente, es decir, en un entorno cercano a la configuración deseada. Sin embargo, en 2013 se publicó un trabajo que trata de conseguir una demostración de estabilidad global para los controladores visuales directos a partir del secuenciamiento de tareas [Li-2013]. En [Pomares-2011] se describe una estrategia de control visual directo para el seguimiento de trayectorias en la imagen, considerando que las referencias visuales en cada instante del seguimiento dependen del tiempo. Esto permite asegurar una estabilidad global a partir de una serie finita de posiciones relativas muy cercanas unas a otras (y, por lo tanto, locales).

El tiempo de adquisición de las imágenes, así como su procesamiento ha ido reduciéndose considerablemente desde la aparición de los sistemas de control visual. Las aplicaciones a las que van destinados los controladores dinámicos basados en visión son aplicaciones que requieren alta velocidad y precisión. Es por ello que se elimina el controlador interno de los robots comerciales para poder actuar directamente sobre los motores de las articulaciones. El retardo que introduciría un sistema de visión lento en un controlador directo de este tipo haría inútil el esfuerzo realizado para su diseño e implementación. Para solucionar este problema, Schuurman et al. proponen el uso de múltiples cámaras fijas observando el área de trabajo del robot [Schuurman-2004]. Las cámaras se sincronizan de forma que siempre se tenga una imagen en un tiempo de muestreo predefinido. El empleo de múltiples cámaras también permite en cierta medida resolver problemas de oclusiones. Fei et al. resuelven el problema de los retardos introducidos por el sistema de visión con un estimador de movimiento basado en redes neuronales que proporcione muestras intermedias [Fei-2006].

Más recientemente, la inclusión de observadores permite predecir el movimiento de los objetos. En [PerezRR-2009], se hace uso de una cámara fija que observa un robot trabajando sobre un plano cuya tarea es realizar el seguimiento y agarre de objetos en movimiento. Los trabajos previos desarrollados por Kelly et al. se amplian en [Kelly-2006]

al generalizar la aproximación de control basado en Jacobiana traspuesta al control visual de manipuladores que no trabajen únicamente en un plano. El robot donde se prueba el controlador propuesto es una muñeca esférica que permite realizar orientaciones en el espacio Cartesiano 3-D, pero no traslaciones. Incrementando el número de grados de libertad controlados, Sauvee et al. diseñan un controlador visual directo basado en imagen para un robot de 6 grados de libertad [Sauvee-2006]. Al obtener los pares y fuerzas que se deben enviar al robot, se resuelven los problemas de visibilidad, saturación de los actuadores y límites de las articulaciones. A partir de una función de coste se aplica un controlador predictivo sobre el modelo no lineal de la dinámica del robot. El sistema se ha probado únicamente en simulaciones.

La mayoría de los trabajos mencionados anteriormente realizan el control con una configuración de cámara fija. Borangiu et al. describen el controlador directo de robots de tipo Scara con cámara en las dos configuraciones: fija y en el extremo [Borangiu-2003]. En [PerezC-2006a] se describe el control visual directo de un robot cartesiano con cámara en el extremo. Al no tener que controlar los grados de libertad de la orientación de la cámara, se reduce la complejidad del controlador, permitiendo resolver el control visual teniendo en cuenta los parámetros dinámicos no lineales del robot. Como se ha comentado al hablar de los controladores visuales basados en posición, los basados en imagen presentan un comportamiento más estable que éstos ante malas calibraciones de los parámetros intrínsecos de la cámara. En este aspecto, destacan los trabajos de Wang et al [Wang-2008], que tratan de resolver el control visual de un robot con 3 grados de libertad a partir de un algoritmo adaptativo que actualiza el valor de la distancia de la cámara al objeto en cada iteración, obteniendo así una matriz de interacción independiente de la distancia. Para obtener esta distancia, en [Cai-2013] se utiliza un par estéreo, que proporciona además nuevas características visuales 3D. La novedad de este último trabajo no es la utilización de un par estéreo para implementar un controlador visual dinámico basado en imagen. Además, proponen la definición de una nueva Jacobiana de imagen para las nuevas características 3D, que también es de rango completo. Siguiendo la línea descrita por Kelly, en [Sequeira-2003] Sequeira et al. proponen una mejora del controlador dinámico propuesto en [Kelly-2000], calculando en cada iteración la rotación de la cámara,

permitiendo su uso en configuraciones *"eye-in-hand"*. Recientemente, Mahony propone un framework para el diseño de nuevos controladores visuales directos basados en imagen [Mahony-2017]. El enfoque propuesto utiliza la estructura de los sistemas hamiltonianos con puertos. Una ventaja clave de este framework es que cada característica de imagen se trata como un puerto de energía separado conectado a la dinámica del sistema. De este modo, las características de imagen separadas pueden ser tratadas como módulos desmontables en el diseño de control que conduce a una serie de ventajas.

En cuanto a las aplicaciones de los sistemas de control visual dinámico basado en imagen, es habitual encontrar en los últimos tiempos trabajos relacionados con vehículos aéreos no tripulados [Jabbari-2014], [Fink-2015], [Serra-2016], [XieH-2016a], [XieH-2016b] [ZhangX-2016]. También se encuentran trabajos donde se tiene en cuenta la dinámica de manos robóticas para definir tareas de agarre y manipulación basándose en la información visual y táctil [Jara-2014].

En esta Tesis se detallará una arquitectura paralela para la implementación hardware basada en FPGA de estos controladores. Dentro del ámbito de los sistemas de control visual basados en imagen es posible encontrar distintos trabajos que tratan de realizar la resolución de redundancia según se lleva a cabo la tarea de control visual. Dentro de estos trabajos cabe citar algunos como [De Luca-2008], [KumarS-2010], [Mansard-2005], [Mukai-2006]. Estos trabajos tratan resolver algunos problemas asociados a la resolución de redundancia como es la estimación de la pseudoinversa de la matriz de interacción, el secuenciamiento de tareas, el uso de la redundancia para evitar oclusiones o limitar el giro de determinadas articulaciones, etc. Sin embargo, ninguno de estos trabajos previos tiene en cuenta la dinámica del robot. Es posible encontrar reducidas investigaciones que utilicen control visual directo con resolución de redundancia, destacando [Sawo-2005], [Trujano-2011]. Los trabajos iniciados en [Pomares-2014] han servido de base para desarrollar algunos de los controladores embebidos que se van a describir en el Capítulo 5 de esta Tesis. En [Pomares-2014], se describe una nueva técnica de control visual dinámico para resolver la redundancia en el nivel de aceleración, garantizando el movimiento correcto, tanto del efector final, como de las articulaciones. El controlador genera los pares

requeridos para el seguimiento de trayectorias en el plano imagen. Sin embargo, para garantizar la aplicabilidad de esta técnica, un recorrido repetitivo seguido por el extremo del robot debe producir un movimiento periódico de la articulación, para lo que se propone la adición al controlador de un término encargado de resolver el caos producido en el sistema. Una mejora de este tipo de controlador se puede ver en una de las aportaciones de esta Tesis [PérezJ-2016], y que se describirá más en profundidad en el Capítulo 5.

Resumiendo, aunque se puede encontrar un gran número de trabajos relacionados con el control visual directo, no es menos cierto que la gran mayoría de las investigaciones en control visual no tienen en cuenta la dinámica del robot. Los artículos que sí incluyen los parámetros dinámicos del robot en el controlador suelen tratar con robots de pocos grados de libertad dado la complejidad de estimar sus parámetros dinámicos. En general, se utiliza una configuración de cámara fija observando tanto el extremo del robot como el objeto sobre el que se quiere posicionar a éste. Sólo unos pocos artículos presentan una aproximación de control visual basado en imagen, y ninguno de los artículos estudiados integran el controlador en una FPGA. Como se describirá a lo largo de esta Tesis, la principal aportación consiste en una arquitectura semiabierta embebida en FPGA para la implementación de controladores visuales basados en imagen. La reconfigurabilidad, el paralelismo y la posibilidad de implementar un sistema de tiempo real proporciona a los controladores implementados en esta nueva arquitectura gran interés científico [Alabdo-2015], [Alabdo-2016a], [Alabdo-2016b], [PérezJ-2015], [PérezJ-2016].

2.4 Implementación de un sistema de control visual directo

En este apartado se describe un ejemplo de implementación de un sistema de control visual directo basado en imagen. Esto permitirá detallar distintos conceptos relacionados con este tipo de sistemas, así como disponer de un punto de partida para las distintas aportaciones descritas en el Capítulo 5.

2.4.1 Matriz de interacción

La matriz de interacción, $\mathbf{L}_s$, es empleada por los controladores visuales basados en imagen con el objetivo de relacionar la velocidad de un punto en el espacio tridimensional con la velocidad del punto correspondiente en el espacio imagen. Considerando una cámara que se encuentra observando un punto ubicado en el extremo de un robot en movimiento, la matriz de interacción ofrecería información acerca de cómo cambia la característica en el plano de la imagen (punto del extremo del robot observado en la imagen) cuando se produce un cambio en el extremo del robot en movimiento:

$$\begin{bmatrix} \dot{f}_x \\ \dot{f}_y \end{bmatrix} = \mathbf{L}_s\left(\boldsymbol{p}_E^C\right)\cdot\begin{bmatrix} \boldsymbol{v}_{tE}^C \\ \boldsymbol{w}_{tE}^C \end{bmatrix} \tag{2.1}$$

donde $\boldsymbol{p}_E^C$ se trata de un punto en el espacio 3D que se moverá con una velocidad de rotación $\boldsymbol{w}_E^C\left(\alpha_E^C, \beta_E^C, \gamma_E^C\right)$, así como también, una velocidad de traslación $\boldsymbol{v}_{tE}^C\left(x_{tE}^C, y_{tE}^C, z_{tE}^C\right)$ ambas con respecto al sistema de coordenadas de la cámara. Además, $\left(f_x, f_y\right)$ son las coordenadas de la característica extraída en la imagen.

En un caso general, considerando $\boldsymbol{p}^C$ la posición de un punto de un objeto observado por la cámara en un espacio de dimensión j, $\boldsymbol{v}^C = \begin{bmatrix} \boldsymbol{v}_t^C & \boldsymbol{w}^C \end{bmatrix}$ será su velocidad respecto al sistema de referencia de la cámara (compuesta por velocidad de traslación y rotación). Se va a representar con **s** un vector de i características observadas en la imagen (medidas en píxeles), mientras que $\dot{\mathbf{s}}$ será la variación de estas características en la imagen. La matriz de interacción se representará como $\mathbf{L}_s(\mathbf{s}, \mathbf{Z})$ y realizará la siguiente transformación:

$$\dot{\mathbf{s}} = \mathbf{L}_s(\mathbf{s}, \mathbf{Z})\cdot \boldsymbol{v}^C \tag{2.2}$$

Donde **Z** es un vector con la profundidad de cada punto característico en la imagen o distancia entre la cámara y el punto correspondiente observado. Desarrollando la expresión anterior, se obtiene la expresión que permite calcular esta matriz de interacción:

$$\dot{\mathbf{s}} = \frac{\partial \mathbf{s}}{\partial \mathbf{r}} \frac{\partial \mathbf{r}}{\partial \mathrm{t}}, \dot{\mathbf{s}} = \begin{bmatrix} \dot{f}_{1x} \\ \dot{f}_{1x} \\ \vdots \\ \dot{f}_{kx} \\ \dot{f}_{ky} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_{1x}}{\partial \mathbf{r}_1} & \cdots & \frac{\partial f_{1x}}{\partial \mathbf{r}_j} \\ \vdots & & \vdots \\ \frac{\partial f_{ky}}{\partial \mathbf{r}_1} & \cdots & \frac{\partial f_{ky}}{\partial \mathbf{r}_j} \end{bmatrix} \cdot \boldsymbol{v}^{\mathrm{C}} \tag{2.3}$$

Donde **r** es la posición tridimensional de la cámara. Como se observa en la expresión (2.3), el número de columnas en la matriz de interacción variará dependiendo de la tarea (se ha considerado un espacio de dimensión j). En general la matriz tendrá 2k filas y j columnas y no se puede asegurar que sea cuadrada.

El valor de la matriz de interacción para el caso de que las características extraídas sean puntos ha sido determinado en numerosos trabajos previos, véase por ejemplo los trabajos de [Chaumette-2006]. En cualquier caso, siguiendo la notación empleada, esta matriz de interacción adquiere el siguiente valor para un espacio de dimensión 6:

$$\mathbf{s} = \begin{bmatrix} \frac{\mathrm{f}_u}{z_{\mathrm{p}}^{\mathrm{C}}} & 0 & -\frac{(f_{\mathrm{x}} - u_0)}{z_{\mathrm{p}}^{\mathrm{C}}} & -\frac{(f_{\mathrm{x}} - u_0)(f_{\mathrm{y}} - v_0)}{\mathrm{f}_v} & \frac{(f_{\mathrm{x}} - u_0)^2 + \mathrm{f}_u^2}{\mathrm{f}_u} & -\frac{\mathrm{f}_u (f_{\mathrm{y}} - v_0)}{\mathrm{f}_v} \\ 0 & \frac{\mathrm{f}_v}{z_{\mathrm{p}}^{\mathrm{C}}} & -\frac{(f_{\mathrm{y}} - v_0)}{z_{\mathrm{p}}^{\mathrm{C}}} & -\frac{(f_{\mathrm{y}} - v_0)^2 + \mathrm{f}_v^2}{\mathrm{f}_v} & \frac{(f_{\mathrm{x}} - u_0)(f_{\mathrm{y}} - v_0)}{\mathrm{f}_u} & \frac{\mathrm{f}_v (f_{\mathrm{x}} - u_0)}{\mathrm{f}_u} \end{bmatrix} \cdot \begin{bmatrix} x_{\mathrm{t}}^{\mathrm{C}} \\ y_{\mathrm{t}}^{\mathrm{C}} \\ z_{\mathrm{t}}^{\mathrm{C}} \\ \alpha^{\mathrm{C}} \\ \beta^{\mathrm{C}} \\ \gamma^{\mathrm{C}} \end{bmatrix} \tag{2.4}$$

En la matriz de interacción aparecen implícitos ciertos parámetros que se pueden extraer a partir del modelo *pin-hole* de una cámara, el cual se encuentra representado en la Figura 2.8. Sea un punto $\boldsymbol{p}^{\mathrm{M}}$ de la escena 3-D en coordenadas del mundo $\boldsymbol{p}^{\mathrm{M}}\left(x_{\mathrm{p}}^{\mathrm{M}}, y_{\mathrm{p}}^{\mathrm{M}}, z_{\mathrm{p}}^{\mathrm{M}}\right)$ y sea $\boldsymbol{p}^{\mathrm{S}}\left(x_{\mathrm{p}}^{\mathrm{S}}, y_{\mathrm{p}}^{\mathrm{S}}\right)$ su proyección en el plano imagen; la proyección $\boldsymbol{p}^{\mathrm{S}}$ en el plano imagen del punto $\boldsymbol{p}^{\mathrm{M}}$ se obtiene como intersección de la línea que une $\boldsymbol{p}^{\mathrm{M}}$ y el centro óptico de la cámara C con el plano imagen. La distancia entre el centro óptico C y el plano imagen se denomina distancia focal, o simplemente focal, f. Al punto que se obtiene como intersección del eje principal z_{C} (línea perpendicular al plano imagen que pasa por el centro óptico C), y al plano imagen se le llama punto principal. Este punto principal (u_0, v_0) representa dos de los parámetros intrínsecos de la cámara. Para obtenerlos, se debe calibrar la cámara, ya sea en una fase previa o mediante una calibración en línea. La

focal, f, es otro de los parámetros intrínsecos que se deben considerar a la hora de modelar una cámara real. Como se acaba de describir, es la distancia entre el sistema de referencia de la cámara y el plano imagen. Normalmente, la distancia focal se suele agrupar con otros dos parámetros que representan el tamaño del píxel: $\left(s_x, s_y\right)$. Así, los parámetros intrínsecos de la cámara se suelen modelar mediante el conjunto $\xi = \{\mathrm{f}_u, \mathrm{f}_v, u_0, v_0\}$, siendo $\mathrm{f}_u = \mathrm{f} \cdot s_x$ y $\mathrm{f}_v = \mathrm{f} \cdot s_y$.

Supóngase el caso en el que el sistema de visión es capaz de extraer k puntos característicos de la imagen. En este caso, el conjunto de características está formado por varios puntos de la imagen expresados en píxeles, $\mathbf{s} = \{f_\mathrm{i}, \mathrm{i} \in 1 \ldots \mathrm{k}\}$, y la matriz de interacción obtenida se puede generalizar para varios puntos de la siguiente manera:

$$\mathbf{L}_\mathrm{s}(\mathbf{s}, \mathbf{Z}) = \begin{bmatrix} \mathbf{L}_{\mathrm{s}1}(f_1, \mathrm{Z}_1) \\ \mathbf{L}_{\mathrm{s}2}(f_2, \mathrm{Z}_2) \\ \vdots \\ \mathbf{L}_{\mathrm{sk}}(f_\mathrm{k}, \mathrm{Z}_\mathrm{k}) \end{bmatrix} \tag{2.5}$$

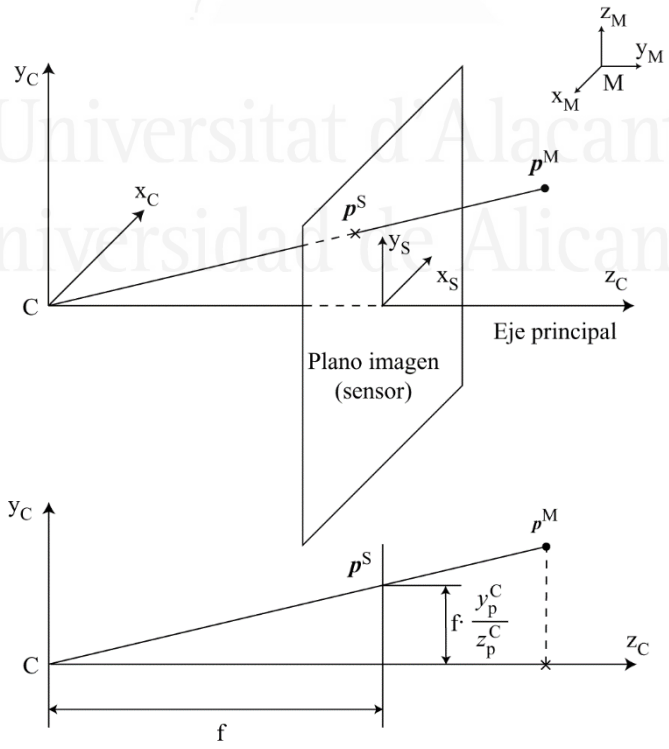


Figura 2.8 Parámetros intrínsecos de la cámara

donde $\mathbf{L}_{si}(f_i, Z_i)$ es la matriz de interacción para el punto característico i. Las dimensiones de la matriz de interacción serán 2k x j siendo k el número de características observadas en la imagen y j los grados de libertad de la tarea.

2.4.2 Controlador directo basado en imagen

Los sistemas de control visual indirectos tienen como objetivo calcular en cada iteración del bucle de control la velocidad en el espacio cartesiano que tiene que seguir el extremo del robot para alcanzar la posición deseada. Para ello, el sistema sólo tiene en cuenta las características cinemáticas del robot, dejando al controlador interno de éste la tarea de asegurar que se consiguen esas velocidades calculando los pares y fuerzas que se deben transmitir a las articulaciones. Un control visual directo elimina este paso, de forma que es el propio controlador visual quien calcula y envía los pares y fuerzas que deben guiar al robot hacia la posición requerida. El resultado es un control mucho más rápido y preciso. 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 [Spong-1989]:

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) \tag{2.6}$$

donde $\mathbf{q} \in \Re^{nx1}$ son las coordenadas articulares del robot, $\dot{\mathbf{q}} \in \Re^{nx1}$ representan las velocidades articulares mientras que $\ddot{\mathbf{q}} \in \Re^{nx1}$ son las aceleraciones articulares (n representa los grados de libertad o número de articulaciones del robot). $\mathbf{M}(\mathbf{q}) \in \Re^{nxn}$ es la matriz de inercia del manipulador que es definida positiva y simétrica, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \in \Re^{nx1}$ es el vector de los pares centrípetos y de Coriolis, $\mathbf{g}(\mathbf{q})$ es la fuerza gravitacional. Finalmente, $\boldsymbol{\tau} \in \Re^{nx1}$ es el vector de los pares aplicados a las articulaciones.

El control visual descrito en [Kelly-2000] 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 [Kelly-2000] viene motivada por la filosofía de control de Jacobiana traspuesta introducido en [Takegaki-1981]. La ley de control propuesta viene dada por:

$$\boldsymbol{\tau} = \mathbf{L}_J{}^T(\mathbf{q},\mathbf{s},\mathbf{Z})\mathbf{K}_P\mathbf{e}_s - \mathbf{K}_D\dot{\mathbf{q}} + \mathbf{g} \tag{2.7}$$

donde $\mathbf{K}_P \in \Re^{2kx2k}$ y $\mathbf{K}_D \in \Re^{nxn}$ son las constantes proporcional y derivativa respectivamente (estas constantes son matrices simétricas definidas positivas elegidas por el diseñador del controlador). Dentro del primer término de (2.7), $\mathbf{e}_s = (\mathbf{s} - \mathbf{s}_d)$ es el error en imagen, siendo $\mathbf{s}_d$ el conjunto de características de la imagen en la posición deseada del robot respecto al objeto y $\mathbf{s}$ el conjunto de las características de la imagen en la iteración actual. $\mathbf{L}_J(\mathbf{q},\mathbf{s},\mathbf{Z}) \in \Re^{2kxj}$ es la matriz de interacción que representa la relación entre la velocidad articular del robot y la variación con respecto al tiempo de las características visuales. El segundo término de (2.7) ($-\mathbf{K}_D\dot{\mathbf{q}}$) se corresponde con la acción derivativa y requiere que se disponga en cada iteración de las velocidades de las articulaciones del robot, $\dot{\mathbf{q}}$. Mientras que el término $\mathbf{g}$ 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, $\mathbf{q}$.

El controlador mostrado en (2.7) presenta una estabilidad asintótica local. En [Kelly-2000] se realiza un estudio completo de esta estabilidad utilizando el método directo de Lyapunov. Este controlador se ha implementado también en la arquitectura abierta embebida en FPGA, como se describirá en el Capítulo 5 de esta Tesis.

2.4.3 Ejemplo de tarea de control visual directo basado en imagen

Para ilustrar lo visto en este apartado supóngase que se dispone de un sistema compuesto por una cámara localizada en el extremo de un robot planar de 2 grados de libertad, y un patrón del cual se extrae un punto visual característico. En la Figura 2.9.a se observa la localización inicial del extremo del robot, mientras que en la Figura 2.9.b se representa la localización deseada. La posición inicial del extremo se marca en amarillo, mientras que la posición final deseada se marca en azul.

Se han elegido los siguientes valores para las constantes $\mathbf{K}_P$ y $\mathbf{K}_D$ del controlador de Jacobiana transpuesta descrito en la Ecuación (2.7):

$$\mathbf{K}_P = \begin{bmatrix} 0.005 & 0 \\ 0 & 1.5 \end{bmatrix} \quad \mathbf{K}_D = \begin{bmatrix} 25 & 0 \\ 0 & 35 \end{bmatrix} \tag{2.8}$$

Este controlador permite realizar un posicionamiento con información visual. Para ello, se almacena la posición del centroide de un círculo en la imagen para la posición final deseada. Este centroide será la referencia del sistema de control. Posteriormente, se desplaza el robot a la posición inicial y el controlador se encargará de, mediante la información de la posición de este centroide, calcular en cada iteración los pares articulares necesarios para llevar a la cámara (situada en una posición relativa fija respecto al efector final del robot) hasta la posición final deseada.

El sistema de control basado en imagen realizará las acciones necesarias sobre el robot para que se produzcan los cambios en las características observadas en la imagen que

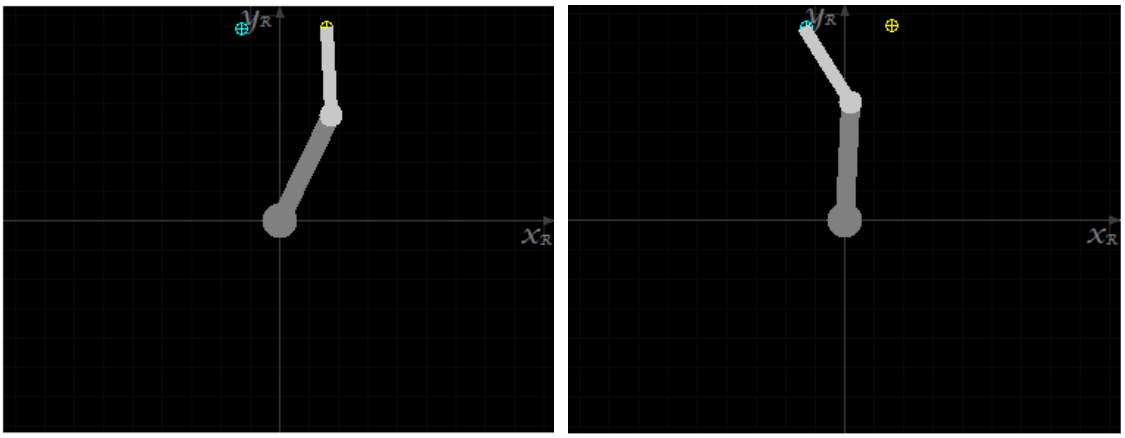
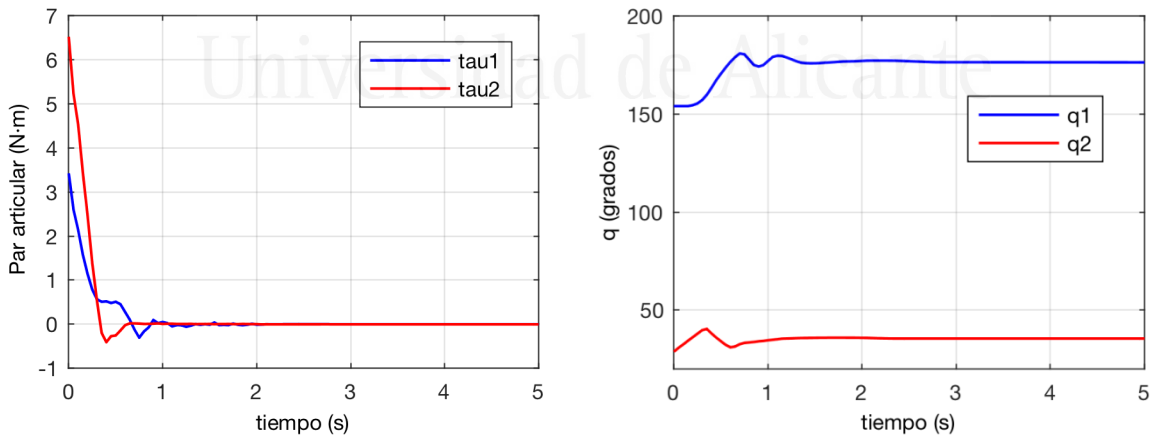


Figura 2.9 (a) Posición inicial del robot **(b)** Posición final

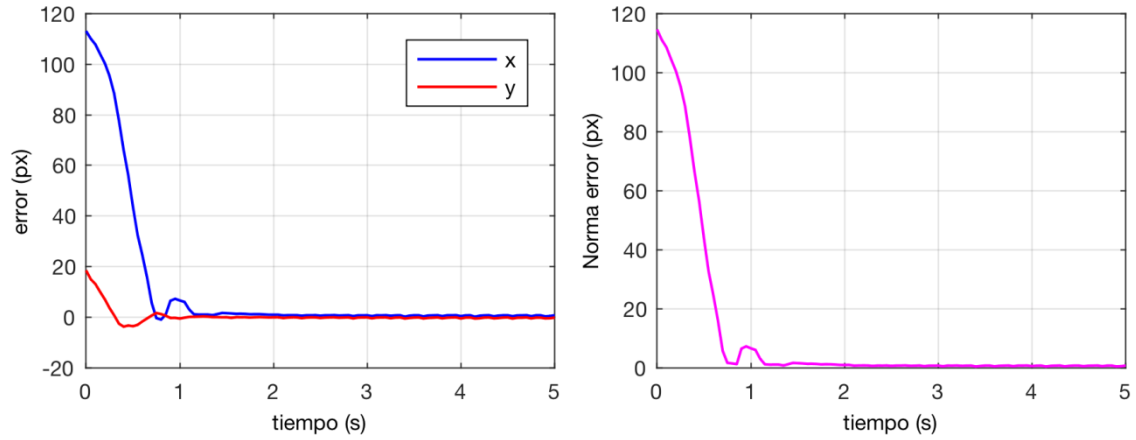
hagan que estas características, **s**, alcancen el valor de las características deseadas, $\mathbf{s}_d$. La Figura 2.10.a muestra la evolución de los pares articulares aplicados al robot durante la tarea de control visual descrita. Para ello, se aplica la Ecuación (2.7) utilizando las ganancias descritas en (2.8). La Figura 2.10.b muestra la evolución durante el posicionamiento de las coordenadas articulares. Se aprecia cómo el sistema evoluciona para q_1 desde poco más de 150 grados hasta casi 180 grados. El comportamiento de la evolución articular es bastante estable y suave.

Con esta acción de control, la evolución del error de la característica visual es la que se muestra en la Figura 2.11.a. El error decrece rápidamente en imagen hasta prácticamente cero en poco más de 1 segundo. En la Figura 2.11.b se muestra la norma del error, compuesta a partir del error en la coordenada X y el error en la coordenada Y del patrón en imagen. Esta gráfica permite observar el comportamiento global en las dos dimensiones de la tarea de un solo vistazo. En la Figura 2.12.a se muestra la trayectoria en el espacio Cartesiano descrita por el extremo del robot para alcanzar la posición deseada desde su posición inicial. En la Figura 2.12.b se muestra la evolución, durante la trayectoria de la cámara, del centro de gravedad del círculo utilizado como característica extraída en la imagen.



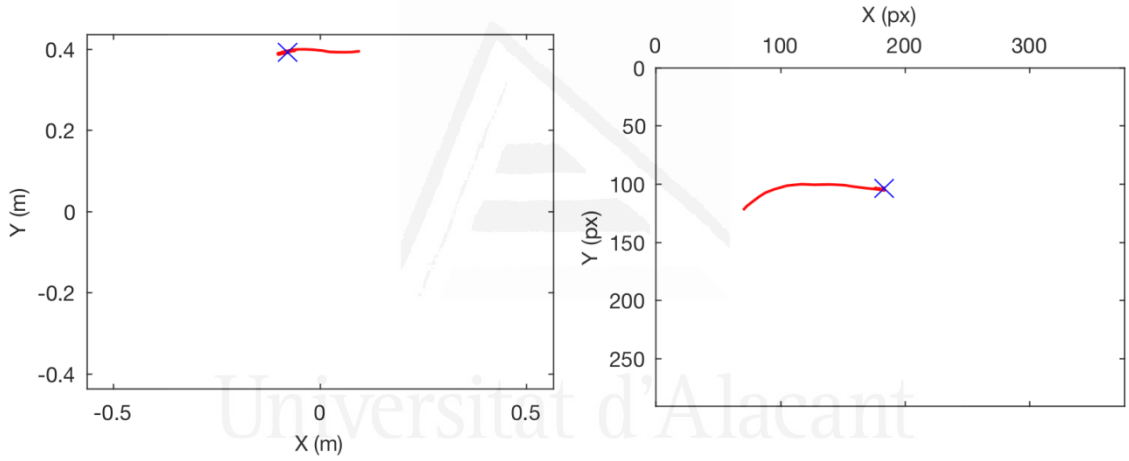
(a) (b)

Figura 2.10 Tarea de control visual directo con el controlador de Jacobiana transpuesta: **(a)** Pares articulares enviados por el controlador de Jacobiana transpuesta **(b)** Coordenadas articulares



(a) (b)

Figura 2.11 Tarea de control visual directo con el controlador de Jacobiana transpuesta: **(a)** Error en imagen **(b)** Norma del error en imagen



(a) (b)

Figura 2.12 Tarea de control visual directo con el controlador de Jacobiana transpuesta: **(a)** Evolución del extremo del robot **(b)** Evolución de la característica visual en imagen

2.5 Conclusiones

A lo largo de este capítulo se ha realizado una revisión de las principales aproximaciones de control visual más relacionadas con las empleadas en la Tesis, lo que ha permitido introducir las principales aportaciones en comparación con el estado del arte actual. Como se ha indicado, el principal objetivo ha sido el desarrollo de una arquitectura basada en FPGA para la implementación de nuevas leyes de control visual directo basadas en imagen. La gran cantidad de información a procesar en la realimentación de los sistemas de control visual, junto a los retardos de los sistemas de captura de imágenes,

representan un grave obstáculo para el diseño de sistemas de control visual directos 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. Gracias a su arquitectura, paralelismo y flexibilidad de configuración, las FPGAs pueden ser empleadas con el propósito de mejorar el desempeño de, tanto los sistemas de captura y procesamiento de imágenes, como los distintos algoritmos de control empleados.

En el siguiente capítulo, Capítulo 3, se describirá en detalle la arquitectura que hace uso de la reconfigurabilidad de las FPGAs para poder aplicarse al control visual directo de distintos robots. Posteriormente, en el Capítulo 4, se muestra la optimización realizada en los procesamientos en imagen embebidos en FPGAs para la obtención de las características visuales empleadas por los controladores. Finalmente, el bloque fundamental de la Tesis finaliza en el Capítulo 5, donde se describen los controladores visuales directos propuestos, así como su implementación en la arquitectura propuesta.

3 Plataforma hardware basada en FPGA para control visual dinámico

Son muchas las ventajas que proporciona el uso de dispositivos FPGA para el diseño e implementación de controladores robóticos. En el capítulo anterior, se ha podido establecer qué aspectos de las FPGAs son más beneficiosos para el desarrollo de nuevos esquemas de control visual, entre los que se puede destacar el comportamiento de estos dispositivos para tratar de manera paralela todo tipo de procesamientos de imagen. Con el objetivo de disponer de una plataforma con la que poder diseñar fácilmente nuevos controladores visuales dinámicos, en este Capítulo se describe una arquitectura flexible y reconfigurable embebida en una FPGA, donde el diseño modular permite separar los componentes dependientes del hardware de los que no dependen de éste.

Para ello, primero se describe el hardware que permite la conexión de la FPGA con los distintos dispositivos que intervienen en una tarea de control visual: robot, cámara, monitor, PC para control de parámetros, etc.

Finalmente, se describe la arquitectura software, enlazando con los dos siguientes capítulos, donde se describirán los distintos elementos que integran los sistemas de visión artificial y control visual.

3.1 Introducción

El sistema de control de movimiento representa un subsistema responsable de la actuación de todo tipo de dispositivos en automatización como la robótica, el mecanizado CNC y las máquinas de montaje de alta velocidad. En estas tareas, mayor velocidad con el mismo error significa mayor productividad. En los últimos años se han propuesto diferentes algoritmos de control. Muchos de estos algoritmos pueden considerarse como casos especiales de la clase de controladores de par. Una manera de clasificar los

esquemas de control de los robots es clasificarlos como controladores directos y controladores indirectos. En los controladores directos, se calculan los pares necesarios para mover el robot en el algoritmo de control mientras que, en los controladores indirectos, se calculan las velocidades articulares que se usan posteriormente como referencia en otros bucles de control responsables de generar los pares correspondientes.

El éxito de los sistemas de control de movimiento depende no sólo del algoritmo de control, sino también de la estructura del hardware de control. Las estructuras de hardware de control utilizadas por los controladores existentes pueden ser construidas utilizando:

1- exclusivamente una DSP de control de movimiento.

2- una DSP junto con un controlador de movimiento ASIC.

3- múltiples DSPs conectadas entre sí.

4- DSP de uso general y recursos de control externos, como una FPGA.

Una combinación de una DSP y una FPGA es la estructura más popular en la actualidad, donde la FPGA implementa las funciones de E/S de acuerdo con los requisitos específicos. El aumento de la productividad, es decir, el rendimiento, es la fuerza motriz detrás de los avances en el diseño de hardware de control de movimiento. Lo que los diseñadores buscan es una mayor velocidad con el mismo error de rendimiento, proporcionando una mayor productividad, todo a bajo coste. Un tiempo de ciclo relativamente bajo conduce a un rendimiento degradado, especialmente a altas velocidades del robot. Además, debido a la limitada potencia computacional de las arquitecturas de control de bajo coste existentes, muchos algoritmos de control avanzados son poco prácticos, ya que la complejidad de estos algoritmos conlleva una baja frecuencia de muestreo, lo que conduce a un rendimiento de control mediocre. Un sistema multi-DSP puede ser una solución a la hora de implementar un modelo complejo del algoritmo de control, pero su estructura incurre en la limitación con respecto al intercambio de datos entre los diferentes DSPs. Por lo tanto, se requiere una nueva arquitectura de controlador, que permita implementar controles avanzados capaces de proporcionar un rendimiento sustancialmente mejorado, especialmente durante movimientos de alta velocidad.

Las FPGAs proporcionan una gran cantidad de recursos para implementar algoritmos digitales complejos, tales como puertas lógicas o bloques RAM. El empleo de E/S cada vez más rápidas y de buses de datos bidireccionales en los diseños de las FPGAs, convierte en un desafío de diseño la verificación de su correcta sincronización de los datos. La planificación de planta (del inglés *Floor Planning*) permite la asignación de recursos dentro de la FPGA para cumplir con estas limitaciones de tiempo. Las FPGAs se pueden utilizar para implementar cualquier función lógica que un ASIC podría realizar. La capacidad de actualizar su funcionalidad y la reconfiguración parcial de una parte de su diseño ofrecen ventajas para muchas aplicaciones.

Algunas FPGAs tienen funciones analógicas además de las funciones digitales. La característica analógica más común es el *Slew Rate* programable en cada pin de salida. También son comunes los osciladores de cristal de cuarzo, los osciladores de resistencia-capacitancia y los lazos de seguimiento de fase PLL (del inglés *Phase-Locked Loops*) con osciladores controlados por voltaje que se utilizan para la generación y gestión del reloj, y para el serializador-deserializador de alta velocidad (SERDES). También son comunes los comparadores diferenciales en los pines de entrada, diseñados para ser conectados a canales de señalización diferenciales. Algunas FPGAs de señales mixtas tienen convertidores analógico-digital (ADC) y convertidores digital-analógico (DAC), con bloques de acondicionamiento de señal analógicos, lo que permite a la FPGA funcionar como un sistema completo en un único chip.

En los últimos años el uso de FPGAs ha aumentado y extendido en sistemas de adquisición de datos, análisis de señales, procesos industriales críticos, etc. Muchos de los algoritmos que antes era prácticamente imposible implementarlos usando procesadores o microcontroladores tradicionales, debido a sus limitaciones tanto en recursos de hardware como en la capacidad computacional, gracias a las FPGAs hoy en día ya es posible implementarlos y obtener el máximo rendimiento de dichos algoritmos.

Con una creciente necesidad de sistemas robóticos avanzados, las FPGAs se están convirtiendo en un dispositivo de diseño a tener muy en cuenta para la próxima generación de robots. Las FPGAs proporcionan una mayor flexibilidad sobre sus alternativas, ya sean microcontroladores (MCUs), procesadores de señales digitales (DSP),

procesadores específicos de aplicación (ASIC) y cualquier combinación de estas arquitecturas.

Tal y como se ha descrito en el Capítulo 2, no es posible encontrar trabajos previos que hagan uso de FPGAs para el control visual directo basado en imagen de robots manipuladores. Sin embargo, sí que están empezando a aplicarse con éxito las FPGAs para la implementación de controladores de robots. Este es el caso de [Zouari-2014] donde se propone una técnica de diseño software y hardware a través de una FPGA para el control de un robot de dos grados de libertad accionado por motores de corriente continua sin escobillas BLDC (del inglés *BrushLess Direct Current*). Este diseño ha demostrado su rapidez, altas prestaciones y bajo coste, características que representan un gran desafío para los sistemas electrónicos de potencia, especialmente en el campo de la robótica. De la misma manera, en [Kim-2007] se presenta la implementación hardware de un controlador PID para el brazo robótico ROBOKER. Un aspecto que habitualmente requiere una gran cantidad de procesamiento en robótica es la implementación de la cinemática del robot controlado. En [SánchezDF-2009] se presenta la implementación basada en FPGA de la cinemática directa de un robot manipulador esférico utilizando unidades de punto flotante. Los resultados de la síntesis muestran que las FPGAs actuales pueden aplicarse para la implementación de la cinemática directa de robots debido, entre otros factores, a la gran cantidad de elementos lógicos disponibles. Los autores muestran una comparación del tiempo de computación necesario para calcular la cinemática directa en la implementación propuesta basada en FPGA frente a una simulación del mismo proceso en el microprocesador PowerPC y Matlab. Los resultados demuestran que la FPGA lo ejecuta de forma mucha más rápida que los otras dos opciones.

El paralelismo y la capacidad de procesamiento de las FPGAs actuales se están empleando actualmente para el control de robots con múltiples ejes. Este es el caso del trabajo presentado en [Ali-2013] en el que se describe la implementación basada en FPGA para el control de los servomotores de un robot de seis patas. El robot hexápodo está dotado de 18 servomotores con 3 servos en cada pata. Cada uno de los servomotores está asignado a una salida PWM y es controlado individualmente por la FPGA. Todas las salidas PWM se sincronizan para conseguir un movimiento suave del hexápodo. Los

controladores se implementan en el chip FPGA Spartan-3 y el diseño de hardware se describe en VHDL. En general, la implementación del controlador en una FPGA ofrece varias ventajas en términos de flexibilidad en el diseño del hardware y la ejecución simultánea de los comandos, comparándolo con un sistema convencional basado en microcontroladores. En [Akkaya-2014] los autores presentan un controlador digital PID paralelo basado en FPGA. El controlador está diseñado e implementado para sistemas de control de múltiples ejes. Se emplea la placa Altera DE0 Nano para el realizar el control de un brazo robótico de 5 grados de libertad. El bloque PID digital, el bloque PWM digital y el bloque de interfaz de entrada analógica están diseñados en una FPGA para controlar la posición articular de cuatro motores. Los resultados demuestran que puede emplearse una FPGA como una solución flexible para sistemas de múltiples ejes.

Los robots humanoides actualmente presentan una gran cantidad de articulaciones con el objetivo de dotar a estos robots de una movilidad lo más similar posible a la del humano. Esto se une a la gran cantidad de información sensorial que habitualmente proporcionan estos robots. De nuevo, es posible encontrar aplicaciones de controladores basados en FPGAs para robots humanoides con el propósito de explotar sus capacidades de procesamiento paralelo y bajo consumo. Por ejemplo, en [Kaminaga-2016] se emplean conjuntamente una MCU (del inglés *Microcontroller Unit*) y una FPGA para realizar el control de los 16 motores de los que dispone un robot humanoide, así como la adquisición de datos sensoriales proporcionados por dicho robot. Este trabajo demuestra que la FPGA permite implementar de forma eficiente controladores para sistemas con gran número de actuadores y sensores. En [Pierce-2014] se propone una estructura modular para construir y desarrollar robots humanoides. Con este diseño se puede controlar múltiples grados de libertad en paralelo desde una única FPGA y es posible emplear algoritmos de control sofisticados aprovechando la gran capacidad computacional de la FPGA integrada en cada módulo y los mínimos retardos de intercomunicación entre sí.

Las capacidades de comunicaciones de las FPGAS y sincronización de datos también es empleada para el control de múltiples robots simultáneamente. En [Chinnaaiah-2015] se detalla el control del comportamiento de dos robots y se describe la estrategia para comunicarse entre sí en tiempo real y su sincronización, con el objetivo de alcanzar una

determinada configuración y evitar obstáculos. Cada uno de los robots usados en este trabajo están equipados con una FPGA (Spartan 6), transceptores PMOD RF2, sensores ultrasónicos y sensores IR. El algoritmo propuesto se implementa eficientemente en hardware y logra el guiado de ambos robots hacia el objetivo, evitando obstáculos, mientras que se comunican entre sí. En la misma línea de implementaciones basadas en FPGAs de robots colaborativos, en [Chinnaaiah-2016a], [Chinnaaiah-2016b] se describen algoritmos para la planificación de trayectos de robots múltiples como el cálculo del camino más corto evitando obstáculos

También es posible encontrar implementaciones basadas en FPGA de controladores cinemáticos cuyo principal objetivo es realizar el guiado de robots. Tal es el caso del algoritmo descrito en [Ardilla-2016] en el que se implementa un procesador software sobre una FPGA para diseñar una arquitectura hardware en forma de sistema en chip programable SoPC para el guiado de robots. En [SánchezLAP-2015] se describe la construcción de un robot móvil sigue-líneas utilizando una cámara conectada a la placa DE0-NANO de Terasic que lleva la FPGA Cyclone IV FPGA de Altera. Los resultados muestran que las FPGAs son muy prometedoras como elementos para el control de robots y para aplicaciones de procesamiento de imágenes. También, dentro del campo de la robótica móvil, en [Pullteap-2016] los autores presentan el diseño de un robot móvil con 21 servomotores, tres sensores infrarrojos y controlado mediante la placa Altera Max II Micro Board que presenta una FPGA. Esta FPGA es programada en VHDL para llevar a cabo tareas de movimientos sencillas evitando obstáculos durante su recorrido. En [Shao-2006] se propone una nueva arquitectura de hardware de control de movimiento trasladando el bucle de control del servo del DSP a una FPGA. Esto está motivado por el hecho de que la lógica hardware de alta velocidad de la FPGA puede mejorar en gran medida la capacidad computacional del controlador al tiempo que reduce la carga del DSP. El algoritmo de control se divide en una parte lineal y otra no lineal, implementada en FPGA y DSP, respectivamente. Los experimentos realizados con un manipulador SCARA de Yamaha demuestran la efectividad del enfoque propuesto, especialmente en movimientos de alta velocidad. El rango de aplicaciones de estos controladores basados en FPGAs para el guiado de robots es cada vez más amplio. Por ejemplo, en [Dagnino-

2016] se presenta el diseño, desarrollo y control de un nuevo sistema robótico basado en FPGA como ayuda durante intervenciones quirúrgicas de fractura de huesos. El objetivo es mejorar la precisión, ergonomía y seguridad del procedimiento quirúrgico tradicional durante este tipo de tratamientos. El manipulador robótico es un robot paralelo de 6 DOF y los experimentos realizados demuestran que el sistema propuesto tiene un nivel ligeramente superior de precisión y repetitividad en comparación con implementaciones clásicas.

Este capítulo presenta una arquitectura embebida reconfigurable para el control visual directo de robots manipuladores basada en FPGA (los principales detalles de esta arquitectura se han publicado en [Alacid-2014], [Alabdo-2014], [Alabdo-2016a]). Esta arquitectura puede ser fácilmente adaptada para el control de cualquier robot articular. Tal y como se describirá a lo largo de este capítulo, la arquitectura propuesta será modular y flexible para poder adaptarse a posibles cambios que puedan producirse como consecuencia de la incorporación o modificación del driver de control, o incluso a cambios en la configuración del sistema de adquisición de datos o su control. Además, cuando se emplean FPGAs para la implementación de sistemas de control, se debe llegar a un compromiso entre precisión del sistema de control y complejidad de la arquitectura hardware. La arquitectura propuesta será evaluada con los controladores visuales descritos en el Capítulo 5.

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. En este Capítulo y a lo largo de esta Tesis, se propone una arquitectura parcialmente abierta para el control visual de robots. En este sentido, la reprogramabilidad y reconfigurabilidad que

proporcionan las FPGAs 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 [Pritschow-1990]. En [Morales-Velázquez-2010] 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 [Hassan-2010], en los que se muestran las características a tener en cuenta en el diseño de arquitecturas de control abiertas. Es de destacar trabajos como [Lalpuriya-2013] en los que se propone el uso de FPGAs para el diseño de controladores de robots debido a su paralelismo y a la flexibilidad que proporcionan. Así, es posible encontrar arquitecturas para el control de robots que integran FPGAs en algún componente de su diseño [Shao-2007]. En [ZhangL-2012] 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 la presente Tesis se muestran los distintos componentes de una arquitectura parcialmente abierta diseñada y optimizada para el control visual directo de robots.

3.2 Arquitectura hardware

La arquitectura hardware propuesta consiste en 3 partes principales: tarjeta FPGA, adaptadores electrónicos y el robot manipulador equipado con una cámara de alta velocidad localizada en su extremo (ver Figura 3.1). Además, se emplea un PC como terminal para configuración de parámetros y visualización de señales.

Con el objetivo de describir la arquitectura reconfigurable propuesta en esta Tesis, a continuación, se indica el hardware utilizado. Se han empleado dos robots para evaluar la arquitectura: el robot Mitsubishi PA-10 de 7 grados de libertad y el robot COOPER de 3 grados de libertad representados en la Figura 3.1. Este robot tiene 3 articulaciones de rotación y cada una de estas articulaciones es accionada por un motor de corriente continua con escobillas (*Brushed DC Motor*) con un reductor planetario, un encoder óptico

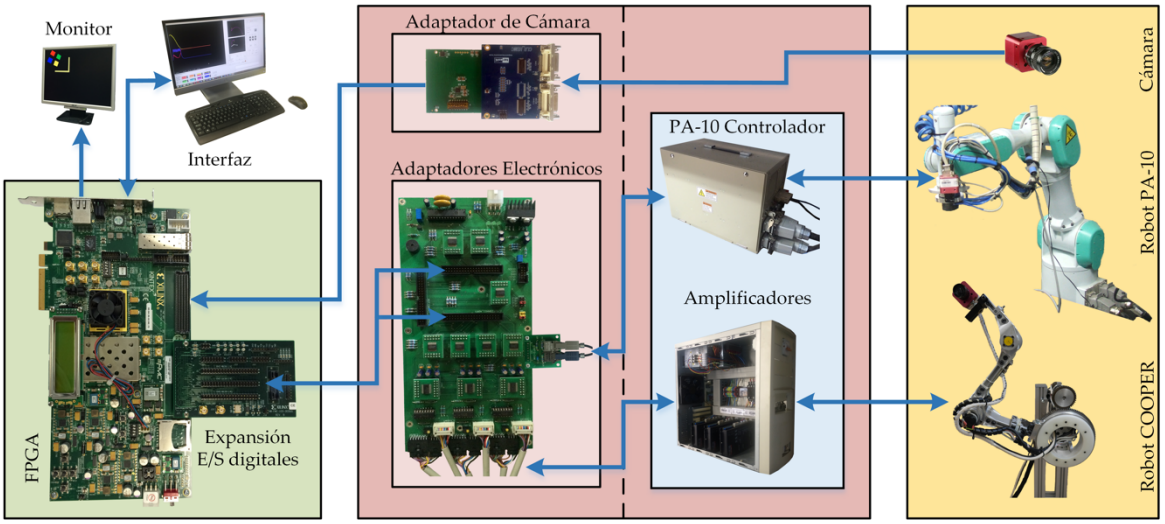


Figura 3.1 Arquitectura hardware propuesta basada en FPGA para el control visual de manipuladores

de cuadratura y un sensor inductivo que actúa como un sensor de final de carrera (las principales características cinemáticas y dinámicas de este robot pueden consultarse en [Pomares-2011a]). Es conveniente resaltar que el diseño de la arquitectura propuesta presenta componentes modulares, con el objetivo de hacerla no dependiente del hardware controlado. Asimismo, se ha empleado una cámara de alta velocidad MV-D752-160 CMOS de Photonfocus. Esta cámara proporciona una resolución de 752 x 582 píxeles y una frecuencia de adquisición de imágenes de hasta 340 fps (del inglés *frames per second*). Se emplea un interfaz *CameraLink* digital para la captura de imágenes.

Para implementar los algoritmos de control y procesamiento de imágenes se utiliza la tarjeta FPGA KC705 de Xilinx (ver Figura 3.2). Estos algoritmos se implementan en VHDL. Esta tarjeta está basada en la FPGA Kintex-7 XC7K325T-2FFG900C de Xilinx. La tarjeta KC705 tiene un oscilador de 200 MHz diferencial LVDS 2.5V. La tarjeta dispone de características habituales a muchos sistemas de procesamiento embebidos, incluyendo una memoria DDR3 SODIMM de 1GB, Entradas/Salidas de propósito general, una interfaz UART, LEDs, switches, etc. Se pueden añadir otras características adicionales usando tarjetas FMCs (del inglés FPGA Mezzanine Cards) conectadas a alguna de las dos expansiones VITA-57-FMC proporcionadas por la tarjeta y denominadas HPC (del inglés *High Pin Count*) y LPC (del inglés *Low Pin Count*).

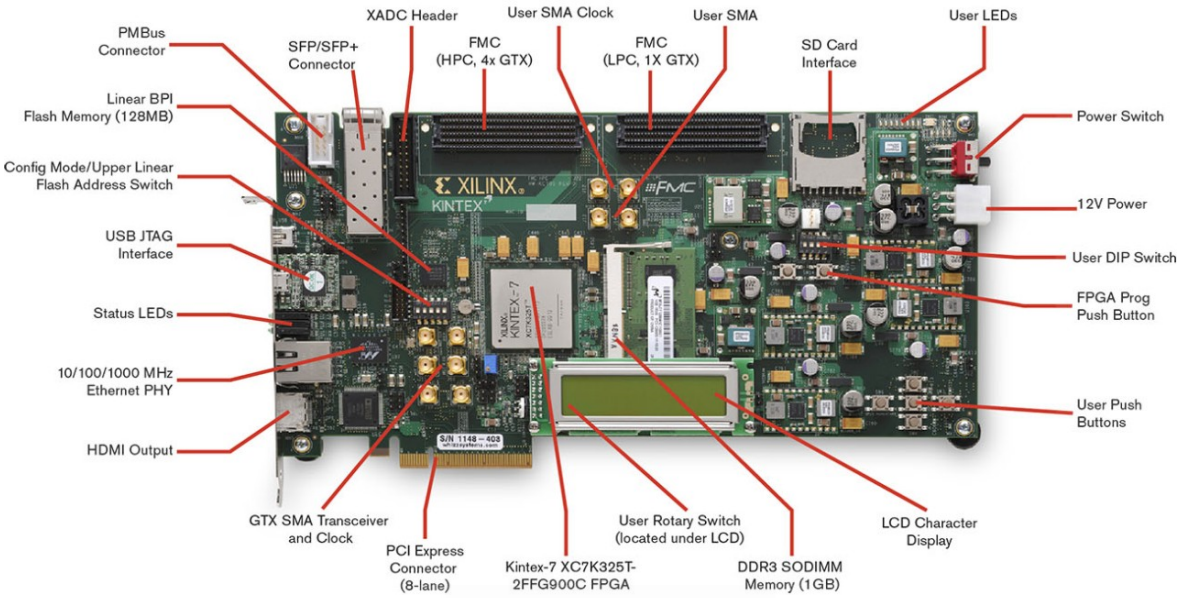


Figura 3.2 La placa de evaluación KC705 de Xilinx (Fuente: [Xilinx-2017])

A continuación, se explica la parte electrónica que se ha diseñado y construido en esta Tesis para poder acceder a los distintos dispositivos que forman parte del sistema de control visual directo propuesto desde la FPGA, como la cámara, los motores, encoders, etc.

3.2.1 Adaptador de la cámara

Con el objetivo de realizar la lectura de imágenes desde la cámara Photonfocus MV-D752-160, se emplea el receptor comercial de *CameraLink* CLR-HSMC, fabricado por Terasic. El CLR-HSMC está diseñado para proporcionar conexiones *CameraLink*, que es un interfaz de comunicación para aplicaciones de visión artificial. Emplea la conexión de alta velocidad HSMC (del inglés *High Speed Mazzanine Card*) como interfaz con otras tarjetas HSMC/HSTC, como las placas de FPGA de Altera. Además, se emplea la tarjeta adaptadora KY-FMC2HSMC de KAYA INSTRUMENTS para habilitar una conexión del conector HSMC en la tarjeta CLR-HSMC al conector FMC LPC en la placa FPGA KC705, tal y como muestra la Figura 3.3.

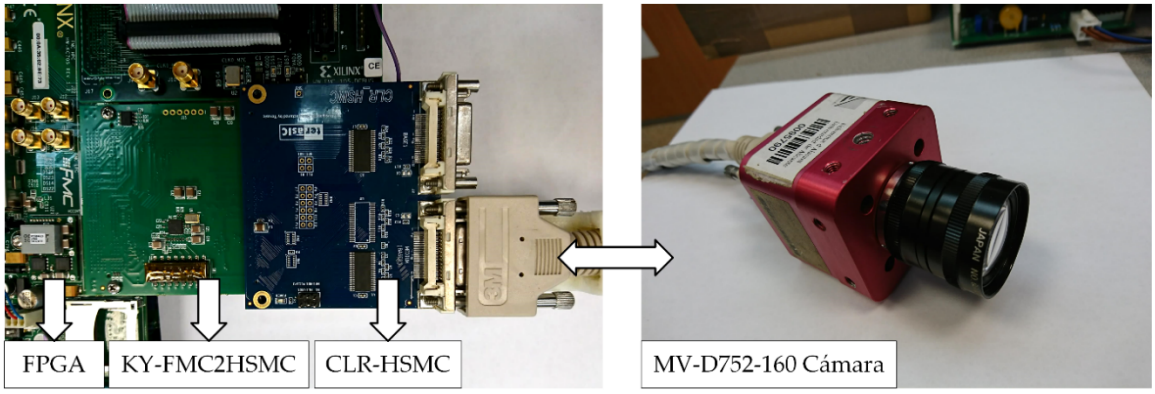


Figura 3.3 Configuración electrónica empleada para conectar la cámara con la FPGA

3.2.2 Interfaz electrónica entre la FPGA y el robot COOPER

El robot COOPER está accionado por 3 motores de corriente continua de 24V, y cada motor está equipado con un encoder óptico de cuadratura que proporciona información imprescindible para calcular la posición, velocidad y aceleración del motor, con el fin de realizar la tarea de control correctamente. Además, cada articulación del robot mencionado tiene un sensor inductivo que actúa como un sensor de final de carrera para determinar la posición inicial de cada articulación.

Para controlar el robot con la FPGA, la placa KC705 empleada en esta Tesis proporciona un gran número de entradas/salidas digitales de propósito general, distribuidas en diferentes clavijas en la placa. Los dos conectores de expansión de E/S más importantes en esta placa son los FMC LPC y FMC HPC (ver Figura 3.2). Estos dos conectores son iguales aparentemente, y la diferencia entre ellos es que el LPC tiene 160 pines disponibles de un total de 400, organizados en 4 columnas con 40 pines en cada una de ellas, como demuestra la Figura 3.4. Mientras que en el conector HPC, como se observa en la Figura 3.5, están disponibles todos los 400 pines del conector, organizados en 10 columnas, cada una con 40 pines.

El conector FMC LPC se reserva en los trabajos de esta Tesis para el módulo receptor de la cámara CLR-HSMC explicado en la Sección 3.2.1. El conector FMC HPC se ha empleado para controlar el resto de dispositivos en el sistema robótico. De los 400 pines del conector HPC destacan 160 E/S digitales de terminación única (*single-ended*) para uso

general, las cuales se pueden usar también como 80 E/S diferenciales. El resto de pines del HPC son 10 transmisores/receptores GTX de alta velocidad, 2 salidas de señal de reloj GTX de alta precisión, 4 señales de reloj diferenciales, 15 tomas de alimentación de diferentes voltajes y 195 puntos de tierra (referencia de voltaje GND).

	K	J	H	G	F	E	D	C	B	A
1	NC	NC	VREF_A_M2C	GND	NC	NC	PG_C2M	GND	NC	NC
2	NC	NC	PRSNT_M2C_L	CLK1_M2C_P	NC	NC	GND	DP0_C2M_P	NC	NC
3	NC	NC	GND	CLK1_M2C_N	NC	NC	GND	DP0_C2M_N	NC	NC
4	NC	NC	CLK0_M2C_P	GND	NC	NC	GBTCLK0_M2C_P	GND	NC	NC
5	NC	NC	CLK0_M2C_N	GND	NC	NC	GBTCLK0_M2C_N	GND	NC	NC
6	NC	NC	GND	LA00_P_CC	NC	NC	GND	DP0_M2C_P	NC	NC
7	NC	NC	LA02_P	LA00_N_CC	NC	NC	GND	DP0_M2C_N	NC	NC
8	NC	NC	LA02_N	GND	NC	NC	LA01_P_CC	GND	NC	NC
9	NC	NC	GND	LA03_P	NC	NC	LA01_N_CC	GND	NC	NC
10	NC	NC	LA04_P	LA03_N	NC	NC	GND	LA06_P	NC	NC
11	NC	NC	LA04_N	GND	NC	NC	LA05_P	LA06_N	NC	NC
12	NC	NC	GND	LA08_P	NC	NC	LA05_N	GND	NC	NC
13	NC	NC	LA07_P	LA08_N	NC	NC	GND	GND	NC	NC
14	NC	NC	LA07_N	GND	NC	NC	LA09_P	LA10_P	NC	NC
15	NC	NC	GND	LA12_P	NC	NC	LA09_N	LA10_N	NC	NC
16	NC	NC	LA11_P	LA12_N	NC	NC	GND	GND	NC	NC
17	NC	NC	LA11_N	GND	NC	NC	LA13_P	GND	NC	NC
18	NC	NC	GND	LA16_P	NC	NC	LA13_N	LA14_P	NC	NC
19	NC	NC	LA15_P	LA16_N	NC	NC	GND	LA14_N	NC	NC
20	NC	NC	LA15_N	GND	NC	NC	LA17_P_CC	GND	NC	NC
21	NC	NC	GND	LA20_P	NC	NC	LA17_N_CC	GND	NC	NC
22	NC	NC	LA19_P	LA20_N	NC	NC	GND	LA18_P_CC	NC	NC
23	NC	NC	LA19_N	GND	NC	NC	LA23_P	LA18_N_CC	NC	NC
24	NC	NC	GND	LA22_P	NC	NC	LA23_N	GND	NC	NC
25	NC	NC	LA21_P	LA22_N	NC	NC	GND	GND	NC	NC
26	NC	NC	LA21_N	GND	NC	NC	LA26_P	LA27_P	NC	NC
27	NC	NC	GND	LA25_P	NC	NC	LA26_N	LA27_N	NC	NC
28	NC	NC	LA24_P	LA25_N	NC	NC	GND	GND	NC	NC
29	NC	NC	LA24_N	GND	NC	NC	TCK	GND	NC	NC
30	NC	NC	GND	LA29_P	NC	NC	TDI	SCL	NC	NC
31	NC	NC	LA28_P	LA29_N	NC	NC	TDO	SDA	NC	NC
32	NC	NC	LA28_N	GND	NC	NC	3P3VAUX	GND	NC	NC
33	NC	NC	GND	LA31_P	NC	NC	TMS	GND	NC	NC
34	NC	NC	LA30_P	LA31_N	NC	NC	TRST_L	GA0	NC	NC
35	NC	NC	LA30_N	GND	NC	NC	GA1	12P0V	NC	NC
36	NC	NC	GND	LA33_P	NC	NC	3P3V	GND	NC	NC
37	NC	NC	LA32_P	LA33_N	NC	NC	GND	12P0V	NC	NC
38	NC	NC	LA32_N	GND	NC	NC	3P3V	GND	NC	NC
39	NC	NC	GND	VADJ	NC	NC	GND	3P3V	NC	NC
40	NC	NC	VADJ	GND	NC	NC	3P3V	GND	NC	NC

Figura 3.4 Disposición de los pines en el conector FMC LPC en la placa KC705

	K	J	H	G	F	E	D	C	B	A
1	VREF_B_M2C	GND	VREF_A_M2C	GND	PG_M2C	GND	PG_C2M	GND	RES1	GND
2	GND	CLK3_M2C_P	PRSNT_M2C_L	CLK1_M2C_P	GND	HA01_P_CC	GND	DP0_C2M_P	GND	DP1_M2C_P
3	GND	CLK3_M2C_N	GND	CLK1_M2C_N	GND	HA01_N_CC	GND	DP0_C2M_N	GND	DP1_M2C_N
4	CLK2_M2C_P	GND	CLK0_M2C_P	GND	HA00_P_CC	GND	GBTCLK0_M2C_P	GND	DP9_M2C_P	GND
5	CLK2_M2C_N	GND	CLK0_M2C_N	GND	HA00_N_CC	GND	GBTCLK0_M2C_N	GND	DP9_M2C_N	GND
6	GND	HA03_P	GND	LA00_P_CC	GND	HA05_P	GND	DP0_M2C_P	GND	DP2_M2C_P
7	HA02_P	HA03_N	LA02_P	LA00_N_CC	HA04_P	HA05_N	GND	DP0_M2C_N	GND	DP2_M2C_N
8	HA02_N	GND	LA02_N	GND	HA04_N	GND	LA01_P_CC	GND	DP8_M2C_P	GND
9	GND	HA07_P	GND	LA03_P	GND	HA09_P	LA01_N_CC	GND	DP8_M2C_N	GND
10	HA06_P	HA07_N	LA04_P	LA03_N	HA08_P	HA09_N	GND	LA06_P	GND	DP3_M2C_P
11	HA06_N	GND	LA04_N	GND	HA08_N	GND	LA05_P	LA06_N	GND	DP3_M2C_N
12	GND	HA11_P	GND	LA08_P	GND	HA13_P	LA05_N	GND	DP7_M2C_P	GND
13	HA10_P	HA11_N	LA07_P	LA08_N	HA12_P	HA13_N	GND	GND	DP7_M2C_N	GND
14	HA10_N	GND	LA07_N	GND	HA12_N	GND	LA09_P	LA10_P	GND	DP4_M2C_P
15	GND	HA14_P	GND	LA12_P	GND	HA16_P	LA09_N	LA10_N	GND	DP4_M2C_N
16	HA17_P_CC	HA14_N	LA11_P	LA12_N	HA15_P	HA16_N	GND	GND	DP6_M2C_P	GND
17	HA17_N_CC	GND	LA11_N	GND	HA15_N	GND	LA13_P	GND	DP6_M2C_N	GND
18	GND	HA18_P	GND	LA16_P	GND	HA20_P	LA13_N	LA14_P	GND	DP5_M2C_P
19	HA21_P	HA18_N	LA15_P	LA16_N	HA19_P	HA20_N	GND	LA14_N	GND	DP5_M2C_N
20	HA21_N	GND	LA15_N	GND	HA19_N	GND	LA17_P_CC	GND	GBTCLK1_M2C_P	GND
21	GND	HA22_P	GND	LA20_P	GND	HB03_P	LA17_N_CC	GND	GBTCLK1_M2C_N	GND
22	HA23_P	HA22_N	LA19_P	LA20_N	HB02_P	HB03_N	GND	LA18_P_CC	GND	DP1_C2M_P
23	HA23_N	GND	LA19_N	GND	HB02_N	GND	LA23_P	LA18_N_CC	GND	DP1_C2M_N
24	GND	HB01_P	GND	LA22_P	GND	HB05_P	LA23_N	GND	DP9_C2M_P	GND
25	HB00_P_CC	HB01_N	LA21_P	LA22_N	HB04_P	HB05_N	GND	GND	DP9_C2M_N	GND
26	HB00_N_CC	GND	LA21_N	GND	HB04_N	GND	LA26_P	LA27_P	GND	DP2_C2M_P
27	GND	HB07_P	GND	LA25_P	GND	HB09_P	LA26_N	LA27_N	GND	DP2_C2M_N
28	HB06_P_CC	HB07_N	LA24_P	LA25_N	HB08_P	HB09_N	GND	GND	DP8_C2M_P	GND
29	HB06_N_CC	GND	LA24_N	GND	HB08_N	GND	TCK	GND	DP8_C2M_N	GND
30	GND	HB11_P	GND	LA29_P	GND	HB13_P	TDI	SCL	GND	DP3_C2M_P
31	HB10_P	HB11_N	LA28_P	LA29_N	HB12_P	HB13_N	TDO	SDA	GND	DP3_C2M_N
32	HB10_N	GND	LA28_N	GND	HB12_N	GND	3P3VAUX	GND	DP7_C2M_P	GND
33	GND	HB15_P	GND	LA31_P	GND	HB19_P	TMS	GND	DP7_C2M_N	GND
34	HB14_P	HB15_N	LA30_P	LA31_N	HB16_P	HB19_N	TRST_L	GA0	GND	DP4_C2M_P
35	HB14_N	GND	LA30_N	GND	HB16_N	GND	GA1	12P0V	GND	DP4_C2M_N
36	GND	HB18_P	GND	LA33_P	GND	HB21_P	3P3V	GND	DP6_C2M_P	GND
37	HB17_P_CC	HB18_N	LA32_P	LA33_N	HB20_P	HB21_N	GND	12P0V	DP6_C2M_N	GND
38	HB17_N_CC	GND	LA32_N	GND	HB20_N	GND	3P3V	GND	GND	DP5_C2M_P
39	GND	VIO_B_M2C	GND	VADJ	GND	VADJ	GND	3P3V	GND	DP5_C2M_N
40	VIO_B_M2C	GND	VADJ	GND	VADJ	GND	3P3V	GND	RES0	GND

Figura 3.5 Disposición de los pines en el conector FMC HPC en la placa KC705

Para tener acceso a los pines del FMC HPC se emplea la tarjeta FMC XM105 Debug Card (Figura 3.6). Esa tarjeta proporciona una serie de clavijas y conectores, de varias posiciones, tipos y tamaños, que facilitan la conexión con las señales de los pines desde y hacia la FPGA. De los cuales se utilizan parte de los pines de los conectores J1 y J3 para controlar los motores, encoders y sensores final de carrera del robot COOPER, y la otra parte se usa para realizar la comunicación con el robot PA-10 y el panel de control que se describe en el Apartado 3.2.2.3.

La Figura 3.7 muestra la placa electrónica que se ha diseñado y construido para conectar la FPGA electrónicamente con los diferentes dispositivos del sistema robótico. A continuación, se detalla el método que hemos seguido para adaptar las E/S de la FPGA con los controladores de los robots mencionados, y se exponen los esquemas de las diferentes partes de esta placa.

3.2.2.1 Adaptador de los motores del robot COOPER

Los motores del robot COOPER son motores de corriente continua de 24VDC con escobillas de Maxon Motors, equipados con encoders ópticos de 3 canales diferenciales y reductoras planetarias (Figura 3.8).

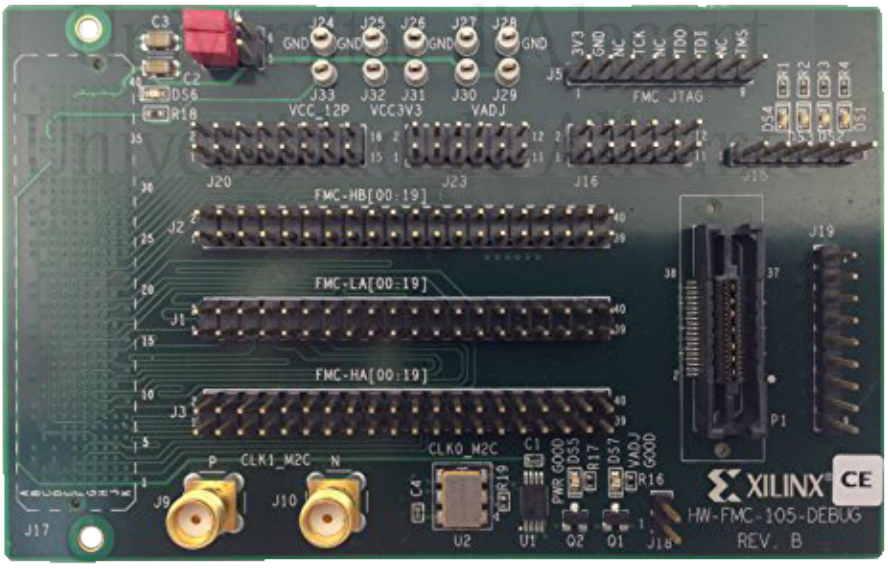


Figura 3.6 Tarjeta XM105 Debug Card

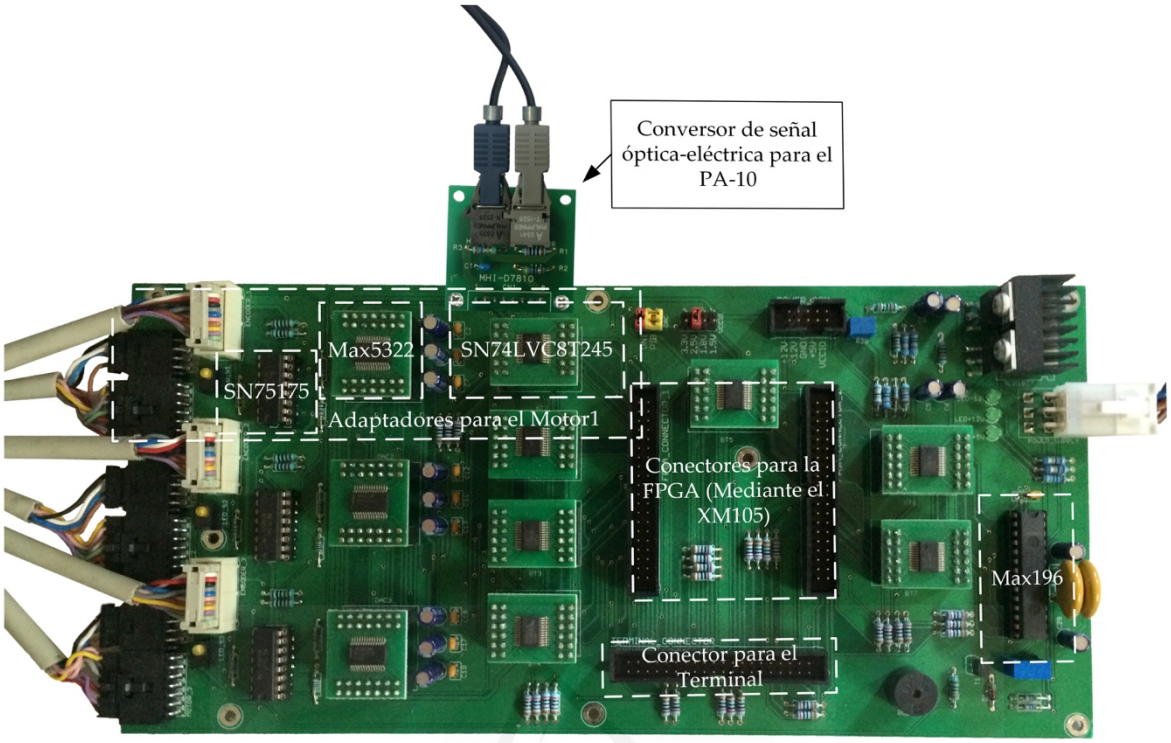


Figura 3.7 Placa electrónica diseñada para controlar los robots con la FPGA

Figura 3.8 Conjunto Motor-Reductora-Encoder de Maxon (Izquierda) y amplificador ADS50/5 usado para controlar los motores (Derecha).

La Tabla 3.1 muestra las características de los 3 motores del robot COOPER, donde M_b es el valor de par nominal, N_b (rpm) es la velocidad nominal, y η_m la eficiencia del motor.

Tabla 3.1 Parámetros de los motores del robot COOPER

	Potencia (W)	M_b (mNm)	N_b (rpm)	η_m (%)
Articulación 1	150	170	6930	91
Articulación 2	90	101	7000	86
Articulación 3	20	30.4	9690	88

La Tabla 3.2 muestra las características de las reductoras del robot COOPER, donde R es la reducción, M_{cont} es el par continuo soportado, M_{int} es el par intermitente máximo soportado por la reductora y η_r la eficiencia de transmisión.

En el caso de la primera articulación, se ha incorporado una primera transmisión con una reducción 110:40 y una eficiencia de 72%. El par final de cada grupo motor-reductor se ha calculado considerando $N_f = Mb \cdot R \cdot \eta_r \cdot \eta_m$. Por lo tanto, los resultados obtenidos para los pares son los mostrados en la Tabla 3.3.

Para gobernar los motores del robot se han utilizado tres amplificadores ADS50-5 de Maxon (Figura 3.8). Estos amplificadores permiten el control a bajo nivel de motores de

Tabla 3.2 Parámetros de reductoras planetarias del robot COOPER

	R	M_{cont} (Nm)	M_{int} (Nm)	η_r(%)
Articulación 1	113:1	15	22	72
Articulación 2	111:1	8	12	70
Articulación 3	159:1	6	7.5	70

Tabla 3.3 Par máximo de las articulaciones del robot COOPER

	Par articular máximo (Nm)
Articulación 1	24.921
Articulación 2	6.749
Articulación 3	2.977

corriente continua de hasta 250W por corriente o par. El ADS50/5 toma como referencia de entrada una señal de voltaje analógica entre -10V y +10V, donde 0V detiene el motor y lo deja en modo frenado, 10V aplica el máximo par (determinado por la configuración de los potenciómetros del ADS50/5) en el sentido horario y -10V lo aplica en el sentido contrario.

La Figura 3.9 muestra el esquema eléctrico utilizado para gobernar un amplificador ADS50/5. Entre las señales de control de amplificador podemos destacar:

- Entradas analógicas. La señal de referencia (*Set Point*) de este amplificador es una señal analógica diferencial aplicada en los pines *+Set Value* y *–Set Value*. Para generar esa señal analógica se ha empleado el DAC (del inglés *Digital-Analog Converter*) MAX5322 de Maxim Integrated. El MAX5322 es un conversor dual de 12 bits de resolución que usa una interfaz de comunicación serie de 10MHz, y proporciona varios rangos de voltaje de salida, entre los que se encuentra el ±10V bipolar que nos interesa para nuestra aplicación.
- Salidas analógicas. Se emplean el *Monitor n* y el *Monitor I*, que proporcionan un valor real de la corriente y velocidad del motor, respectivamente. Estas dos señales son muy importantes para tener realimentación en los bucles cerrados de control de par o velocidad del motor. Estas señales son analógicas y tienen rango de tensión de -10V a +10V. Con el fin de leer estas señales con la FPGA se ha empleado el ADC (del inglés *Analog-Digital Converter*) MAX196 de Maxim Integrated. El MAX196 es un conversor ADC de 12 bits de resolución que usa una interfaz de comunicación paralela y trabaja con varios rangos de voltaje de entrada, entre ellos el ±10V bipolar que se necesita en la tarea de lectura de información desde el robot.
- Entradas digitales. El ADS50/5 tiene varias entradas digitales entre las cuales solo es necesaria para el control del robot la señal de *Enable*, para activar y desactivar el amplificador por software.
- Salidas digitales. La señal *Ready* es una señal digital que refleja el estado del amplificador en cada momento y ayuda a detectar fallos en el funcionamiento del sistema robótico.

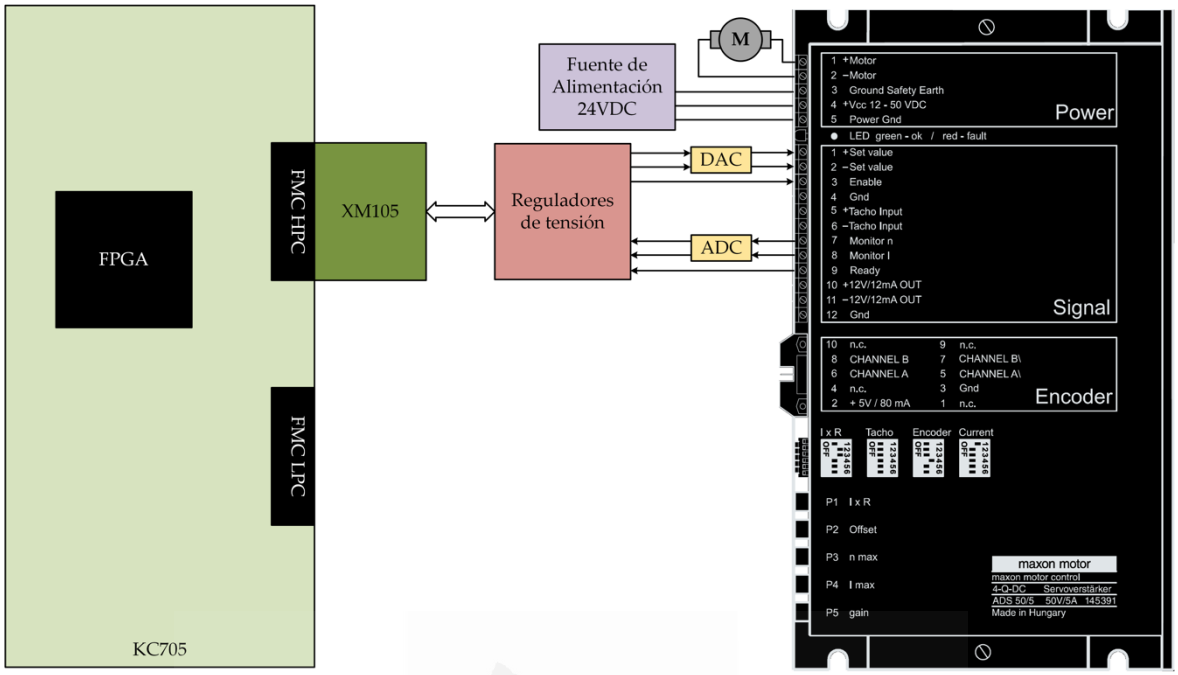


Figura 3.9 Esquema del circuito electrónico para controlar un motor del robot COOPER con la FPGA

Los niveles de voltaje de las E/S en las nuevas generaciones de FPGAs son diferentes y cada vez son menores como, por ejemplo, 1.2V, 1.5V, 1.8V, etc. Sin embargo, muchas FPGAs tienen estándares de salida de voltaje programables y pueden ser configuradas para que se adapten con los estándares de voltaje de los dispositivos externos conectados a la FPGA. Las nuevas generaciones de FPGAs tienen múltiples bancos de E/S, cada banco se alimenta por un carril de alimentación VCCIO independiente, permitiendo que las E/S funcionen a través de múltiples estándares de señal. Los pines de E/S con estándares compatibles se agrupan en el mismo banco de E/S físico, por lo que todos los pines de E/S que pertenecen al mismo banco deben estar configurados para operar con el mismo voltaje. La Tabla 3.4 lista los bancos de E/S disponibles en la FPGA Kintex7, así como el estándar de voltaje asignado a cada banco. Algunos pines de E/S de usuario de la FPGA utilizados en esta Tesis comparten el mismo banco de E/S con periféricos ya incorporados en la placa KC705, tales como la memoria RAM DDR3, el oscilador de reloj, el conector PCI Express, etc. Esto obliga a que todas las E/S de dicho banco trabajen con la tensión de trabajo del periférico ya conectado con el mismo banco.

Tabla 3.4 Los bancos de E/S en la FPGA Kintex7 y sus voltajes

Banco de FPGA	Estándar de Voltaje	Voltaje
Bank 0	VCC2V5_FPGA	2.5V
Bank 12	VADJ_FPGA	1.8V, 2.5V, 3.3V
Bank 13	VADJ_FPGA	1.8V, 2.5V, 3.3V
Bank 14	VCC2V5_FPGA	2.5V
Bank 15	VCC2V5_FPGA	2.5V
Bank 16	VADJ_FPGA	1.8V, 2.5V, 3.3V
Bank 17	VADJ_FPGA	1.8V, 2.5V, 3.3V
Bank 18	VADJ_FPGA	1.8V, 2.5V, 3.3V
Bank 32	VCC1V5_FPGA	1.5V
Bank 33	VCC1V5_FPGA	1.5V
Bank 34	VCC1V5_FPGA	1.5V

Las señales digitales del ADS50/5, MAX5322 y MAX196 operan con diferentes niveles de tensión. Por ejemplo, para activar el ADS50/5 hay que poner la señal *Enable* a nivel alto (*high* o 1 Lógico), y según la hoja de datos del amplificador, la tensión mínima que pone el *Enable* a nivel alto es +4V, un valor que es imposible proporcionar con una salida digital de la FPGA utilizada. Para adaptar los niveles de tensión de la FPGA a los distintos dispositivos del sistema se ha empleado el conversor DC-DC de voltaje SN74LVC8T245 de Texas Instruments. El chip SN74LVC8T245 en un transceptor de bus de 8 bits que usa dos carriles de alimentación separados e independientes, lo que permite usarlo como un regulador de tensión bidireccional rápido. Cada chip puede manejar hasta 8 señales de manera independiente.

3.2.2.2 Adaptador de los encoders del robot COOPER

Los encoders del robot COOPER son encoders ópticos de cuadratura. Cada encoder tiene 3 canales diferenciales A, B e I compatibles con el estándar TTL (es decir, operan con niveles de tensión de +5V). Los canales A y B están desfasados 90 grados, con lo cual no sólo permiten calcular el ángulo de giro del motor sino también el sentido de giro. El canal I proporciona un pulso por vuelta, lo que ayuda a detectar errores en el recuento de pulsos y rectificarlo. La Tabla 3.5 muestra las características de los tres encoders que incorpora COOPER.

Tabla 3.5 Parámetros de los encoders de los motores del robot COOPER

	Pulsos Por Vuelta	Velocidad Máxima (RPM)
Articulación 1	1024	18750
Articulación 2	512	18750
Articulación 3	512	37500

La Figura 3.10 muestra el esquema electrónico realizado para leer las señales de un encoder con la FPGA. Se ha utilizado el receptor de línea diferencial SN75175 de Texas Instruments para convertir las señales del encoder de señal diferencial a señal de terminación única (*single-ended*), con el objetivo de ahorrar la mitad de las entradas de la FPGA requeridas para hacer la interfaz con el encoder. El SN75175 tiene 4 canales, lo que permite manejar 4 señales diferenciales simultáneamente. Este chip requiere también una fase de conversión de voltajes entre la FPGA y el SN75175 por el mismo motivo que se ha detallado en la sección 3.2.2.1 donde se hace uso del chip SN74LVC8T245 como regulador de tensión.

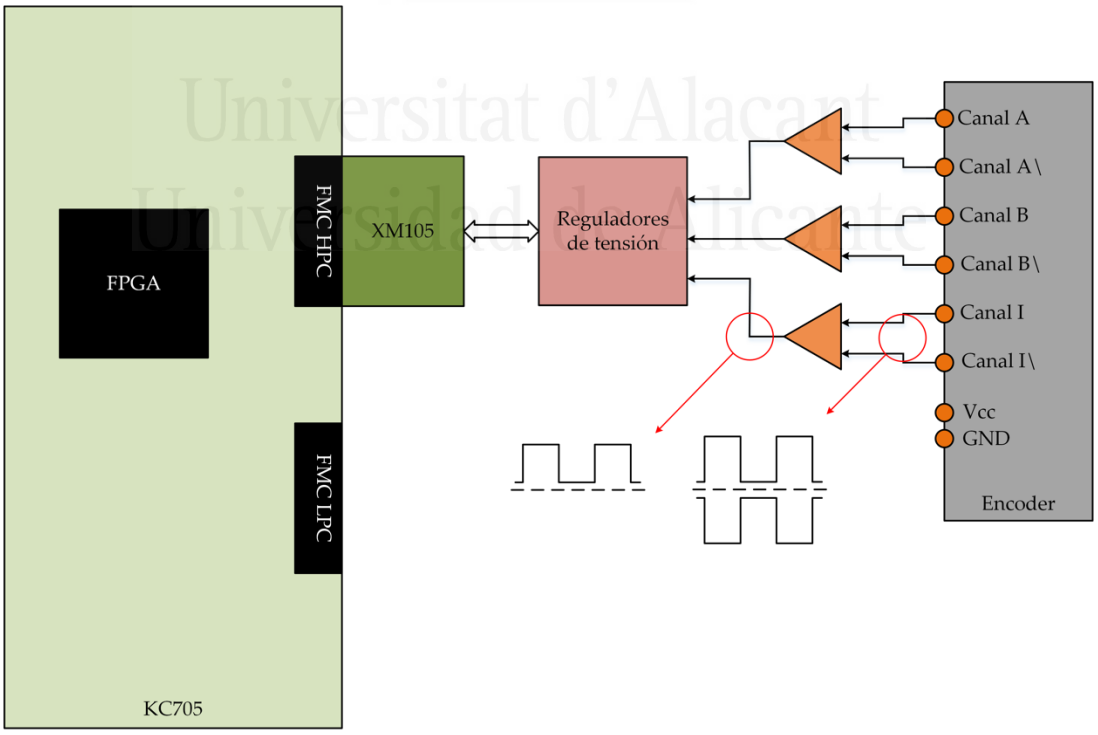


Figura 3.10 Esquema del circuito electrónico para la conexión del encoder de un motor con la FPGA

3.2.2.3 Otros equipamientos electrónicos

Cada articulación del robot COOPER está equipada con un sensor inductivo que permite determinar la posición inicial de cada articulación, y ayuda a situar el robot en el estado de inicio (*Home).* La salida de los sensores está conectada con la FPGA a través de un SN74LVC8T245 para regular la tensión de salida y también como una protección para la entrada de la FPGA.

Para facilitar el manejo del robot COOPER y algunos parámetros del sistema de control visual, se ha creado el terminal que puede verse en la Figura 3.11. La función de cada elemento en este terminal está programada en la FPGA de la siguiente forma:

- Botón amarillo. Determina si se va a manejar el robot con el terminal, o con el interfaz software realizado (se detalla más adelante en este capítulo) para controlar y monitorizar todos los detalles del sistema robótico desde un PC.
- Selector de 3 posiciones Modo de Control. Elige entre mover los motores de forma manual a través de los botones de Motor1, Motor2 y Motor3, o aplicar una ley de control ya programada en la FPGA para realizar la tarea de control visual deseada.
- Selector de 2 posiciones Controlador. Permite elegir aplicar la ley de control de la Jacobiana transpuesta o un controlador visual directo de los propuestos en el Capítulo 5.
- Encoder giratorio. Se emplea para cambiar el valor de algunos parámetros del sistema, dependiendo de la función mostrada en la pantalla LCD, como puede ser el nivel del umbral para la fase de binarización en el módulo de visión, la velocidad articular de los motores cuando se desea moverlos en modo manual, etc.
- Botones azul y verde. Se emplean para elegir los parámetros que interesa monitorizar en la pantalla LCD como, por ejemplo: posición, velocidad y aceleración de las articulaciones de cada motor, el par que genera la ley de control, las características extraídas en el módulo de visión, el umbral de la binarización de imagen, el error y norma del error en el espacio imagen, etc.

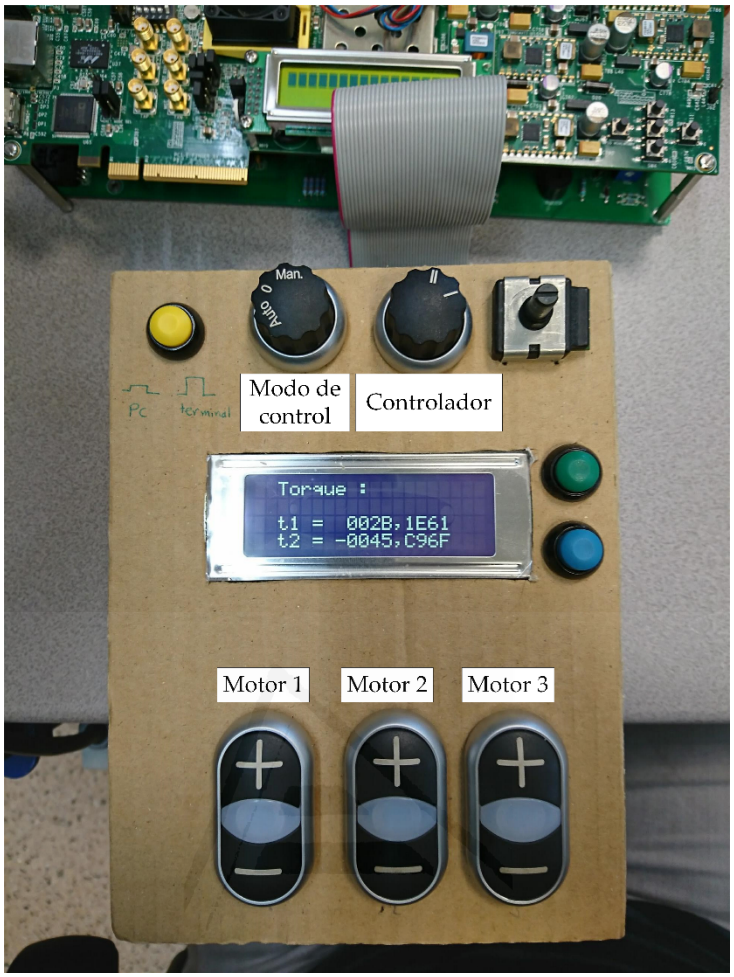


Figura 3.11 Terminal de control

3.2.3 Adaptador de comunicación para el robot Mitsubishi PA-10

El robot Mitsubishi PA-10 es un robot manipulador de 7 grados de libertad. Los motores del PA-10 son motores de corriente continua sin escobillas (*Brushless DC Motor* o *BLDC*) de 100VDC. Cada articulación incorpora una reductora harmónica y un encoder tipo *Resolver*. El controlador del PA-10 incluye dos placas electrónicas principales:

- La placa de potencia: suministra la tensión y potencia adecuada para todas las partes del controlador.
- El Servo Driver: incorpora un módulo amplificador IGBT 6MBP20RH060 para cada motor, así como toda la electrónica necesaria para accionar los motores y obtener datos de los Resolvers y el resto de sensorización del robot y la comunicación con el controlador de alto nivel (el PC o lo que se llama en el manual *Upper Controller*).

El Servo Driver se comunica con el controlador a través del protocolo serie ARCNET y emplea dos hilos de fibra óptica para enviar y recibir las tramas de datos. Para realizar la comunicación entre el PA-10 y la FPGA, se ha añadido a la placa adaptadora un conversor de señal (ver Figura 3.7) que convierte, tanto la señal óptica recibida del Servo Driver a señal digital electrónica compatible con la FPGA, como la señal digital de la FPGA a señal óptica para enviarla al Servo Driver.

3.3 Arquitectura software

Esta sección define la organización de la arquitectura propuesta para la implementación del sistema de control visual basado en FPGA propuesto en esta Tesis. Esta arquitectura propone una implementación software basada en FPGA dividida en distintos módulos, con el objetivo de hacer el sistema flexible, modular y aplicable a distinto tipos de cámaras y robots. Dichos módulos pueden ser genéricos o dependientes de la cámara, o del robot a controlar, y realizan distintas funciones sincronizadas que, además, pueden ejecutarse en paralelo aprovechando las capacidades de procesamiento paralelo de la FPGA. La Figura 3.12 representa un esquema global del sistema de control visual y la arquitectura software propuesta. La arquitectura software embebida en la FPGA está implementada totalmente en VHDL, y está dividida en bloques funcionales independientes que trabajan juntos de forma sincronizada. La idea principal de implementar un sistema modular es hacer un sistema flexible y genérico, en la medida posible. El diseño modular permite modificar cualquier módulo sin que esto afecte al resto de módulos del sistema. Por ejemplo, en la arquitectura de la Figura 3.12, para usar otro tipo de cámaras, sólo es necesario modificar el módulo Digitalizador de Video, que es el encargado de leer los datos de la imagen. Sin embargo, el módulo Algoritmos de Visión es genérico, y no depende directamente de la cámara. Y lo mismo pasa con los robots, cambiar el tipo de motor o encoder de una articulación supone adaptar sólo el módulo que se comunica directamente con este motor o encoder, es decir, el módulo Interfaz del Amplificador o el Calculador de Posición, Velocidad y Aceleración respectivamente sin necesidad de modificar los módulos independientes del hardware, como el módulo Ley de Control, por ejemplo.

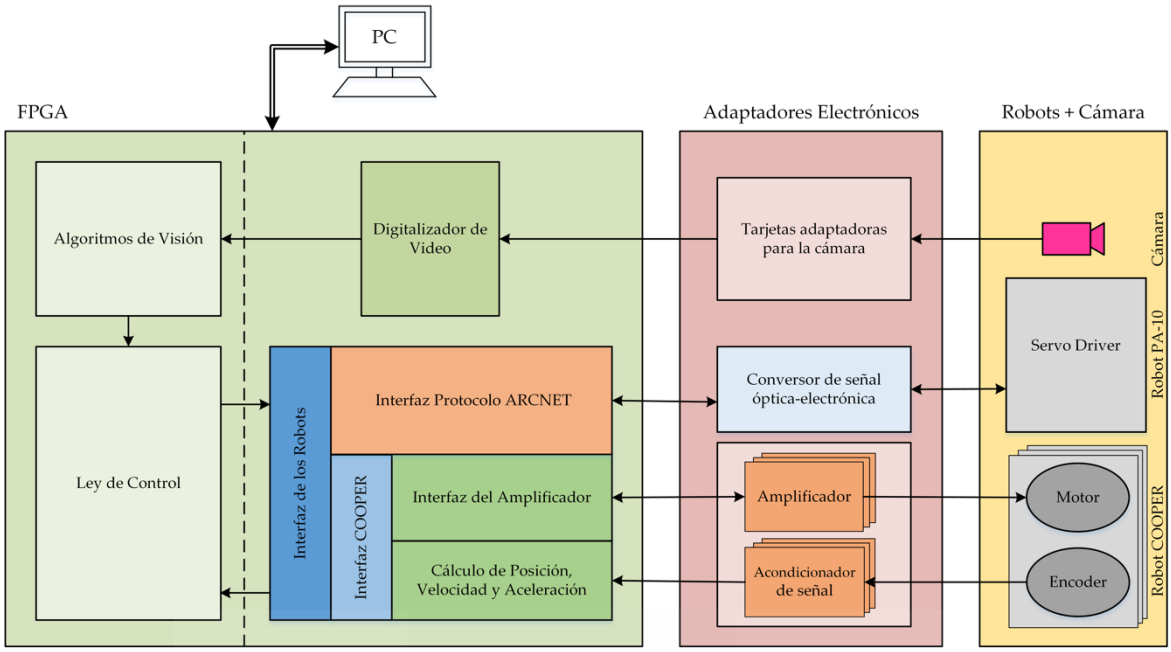


Figura 3.12 Arquitectura software embebida propuesta para el desarrollo de sistemas de control visual directo de robots manipuladores basado en imagen

En la sección 3.2 se ha explicado el hardware empleado en el sistema de control visual, así como el desarrollo de los adaptadores electrónicos realizados para lograr un control completo de todos los dispositivos del sistema desde la FPGA. A continuación, se detalla el funcionamiento del software embebido y la función que cumple cada módulo.

3.3.1 Módulos dependientes del hardware

Son aquellos módulos que se encargan de controlar los dispositivos hardware externos directamente desde el software de la FPGA. En la estructura propuesta en la Figura 3.12 es posible destacar dos módulos principales de este tipo que son: el módulo Digitalizador de Video y el módulo de Interfaz de los Robots.

3.3.1.1 Digitalizador de Video

Este módulo realiza la lectura de los datos de la imagen capturada por la cámara. El módulo, obviamente, depende del protocolo usado por la cámara empleada en la aplicación de control visual (por ejemplo, *CameraLink*, GigE Ethernet, etc.). La cámara utilizada en esta Tesis es la MV-D752-160 CMOS de Photonfocus, que emplea una comunicación *CameraLink* para la transmisión de datos. Los parámetros de la cámara (fps,

tiempo de exposición, resolución, etc.) son configurables a través de la comunicación serie integrada en el estándar *CameraLink*. El módulo Digitalizador de Video tiene que saber estos parámetros a priori, para recibir los píxeles correctamente, y realizar así la sincronización con el resto de módulos de visión de manera segura. La cámara mencionada arriba tiene una peculiaridad, que en cada ciclo de reloj de la cámara (PCLK) envía dos píxeles juntos al Digitalizador de Video. Esto demuestra que es casi imposible diseñar este módulo de forma genérica, porque las características de las cámaras (protocolo de comunicación, resolución, formato de datos de píxeles, etc.) son muy amplias, y cada vez son más sofisticadas. El Digitalizador de Video debe pasar al módulo Algoritmos de Visión las características de la imagen (resolución, bits por pixeles, etc.) con las que va a trabajar la cámara (sólo una vez, justo después de poner el sistema en marcha).

Para la cámara utilizada en esta Tesis, la tarea implementada en VHDL sobre la FPGA consiste en recibir los píxeles de la imagen (2 en dada PCLK), y detectar cuándo se inicia la captura de una imagen nueva y cuándo termina, con la señal FVAL, y cuando se inicia la transmisión de una línea nueva y cuando termina, con la señal LVAL (ver Sección 4.2.1). Con estas señales, en cada PCLK, el Digitalizador de Video envía al módulo Algoritmos de Visión los dos píxeles y sus coordenadas (x, y) en la imagen que está siendo capturada.

3.3.1.2 Interfaz de los Robots

Este módulo se encarga de comunicar directamente con los robots, a través de la electrónica necesaria. Igual que en el caso de la cámara, hoy en día hay una gran cantidad de sistemas robóticos que emplean gran variedad de comunicaciones, motores, encoders, sensores, etc. Por lo tanto, este módulo es dependiente del hardware externo, y no puede ser implementado de forma genérica.

La tarea de este módulo se resume en leer, o calcular, el estado de cada articulación del robot (posición o ángulo de giro, velocidad, aceleración, etc.), y enviarlo al módulo Ley de Control. De vuelta recibe de este último el valor de par que se debe aplicar a las articulaciones del robot a través de su controlador o amplificadores.

3.3.1.2.1 Interfaz del robot COOPER

Se realiza un control a bajo nivel con el robot COOPER, lo que significa que todos los parámetros de las articulaciones se tienen que leer a partir de los pulsos de los encoders. Al mismo tiempo, el par que hay que aplicar al motor se tiene que ejecutar a bajo nivel con los amplificadores ADS50/5 indicados en la Sección 3.2.2.1. La interfaz de COOPER consiste principalmente en dos submódulos que son:

1. Interfaz del Amplificador. Este módulo es responsable de la conversión de los pares obtenidos por el módulo Ley de Control. El resultado es un conjunto de señales adecuadas que controla los drivers de los motores. Así, este tipo de señales depende del tipo de señal de entrada que emplean estos drivers como referencia. La gran mayoría de drivers de amplificación para motores DC emplean la técnica PWM como señal de entrada, debido a su simplicidad. Sin embargo, hay otros drivers que usan señales de entrada diferentes, como pueden ser señales analógicas de corriente o voltaje (este es el caso de los motores del robot empleado para validar la arquitectura propuesta en esta Tesis). Para gobernar el ADS50/5, como se observa en la Figura 3.13, se ha implementado en este módulo 3 submódulos (uno para cada motor, con el objetivo de promover la estructura modular, y poder añadir más motores al sistema con simplemente instanciar este submódulo las veces que se necesiten). Cada submódulo contiene el software necesario para realizar la comunicación serie SPI con el DAC MAX5322 que genera la señal de referencia analógica *Set Value* (ver Figura 3.9). También tiene implementado el software que maneja las señales digitales del ADS50/5 y las del ADC196, que permiten leer el par y velocidad reales del motor, y emplearlos para hacer un bucle de control cerrado.
2. Cálculo de Posición, Velocidad y Aceleración. Este módulo proporciona la posición, velocidad y aceleración actual de los motores del robot para su posterior uso en los algoritmos de control. Los sistemas robóticos pueden emplear diferentes tipos de encoders dependiendo de la tarea a realizar. Entre ellos destacan el

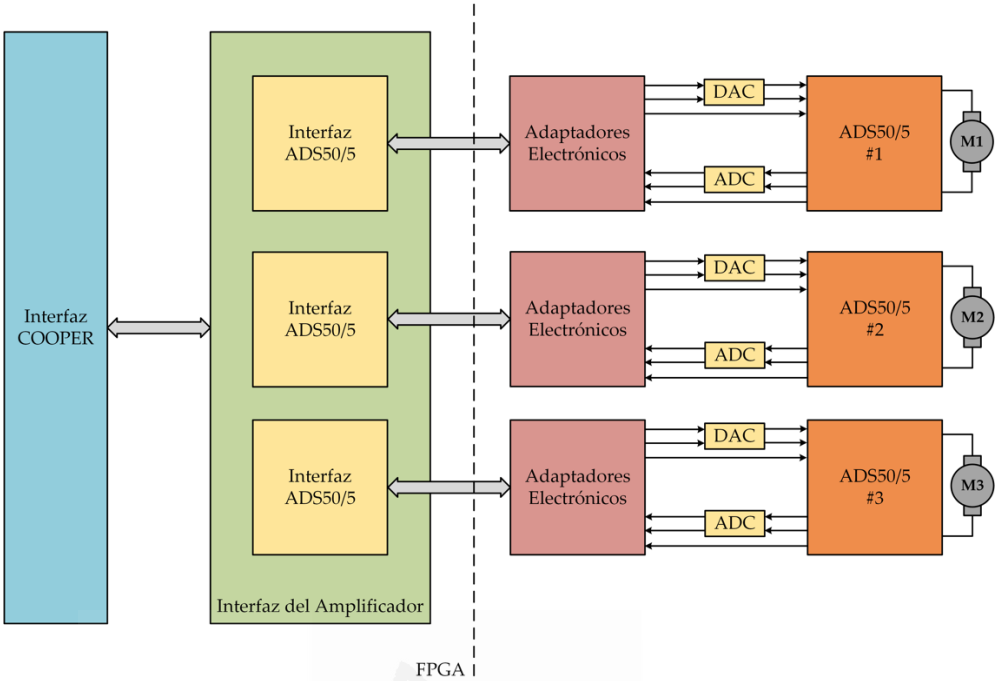


Figura 3.13 Arquitectura del módulo Interfaz del Amplificador

encoder de cuadratura rotatorio y el encoder absoluto rotatorio. Además, pueden emplear tecnología conductiva, óptica, magnética, etc. El robot COOPER emplea 3 encoders ópticos de cuadratura. Cada encoder tiene 3 canales: A, B e I. La Figura 3.14 muestra la estructura modular de este bloque, que contiene un submódulo para cada encoder. Así que, para añadir otro encoder al sistema robótico, resulta sencillo instanciar otro submódulo capaz de leer las señales del encoder añadido sin afectar al resto de módulos.

La Figura 3.15 muestra un esquema funcional del submódulo Interfaz Encoder implementado para cada articulación de COOPER en la FPGA. Este esquema consiste en tres fases: tres filtros de ruido eléctrico (uno para cada canal), un decodificador de pulsos en cuadratura y, finalmente, tres bloques para calcular la posición, velocidad y aceleración. El decodificador de pulsos usa las señales A y B para generar un tren de pulsos y detectar el sentido de giro al estar las señales de A y B desfasadas 90 grados. La posición angular de la articulación se calcula finalmente contando los pulsos del tren de pulsos, teniendo en cuenta la dirección de giro y el número de pulsos por vuelta, y el ratio

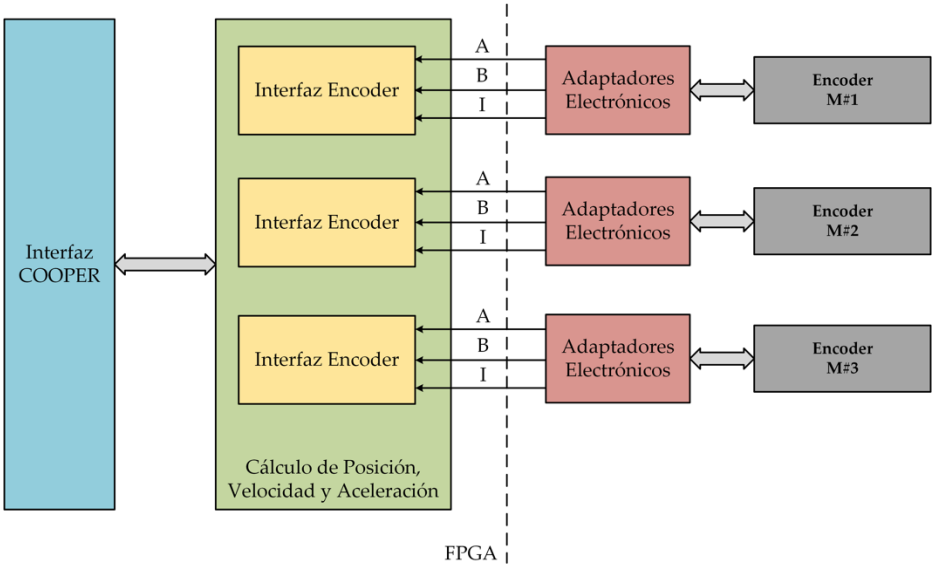


Figura 3.14 Arquitectura del módulo de Cálculo de Posición, Velocidad y Aceleración

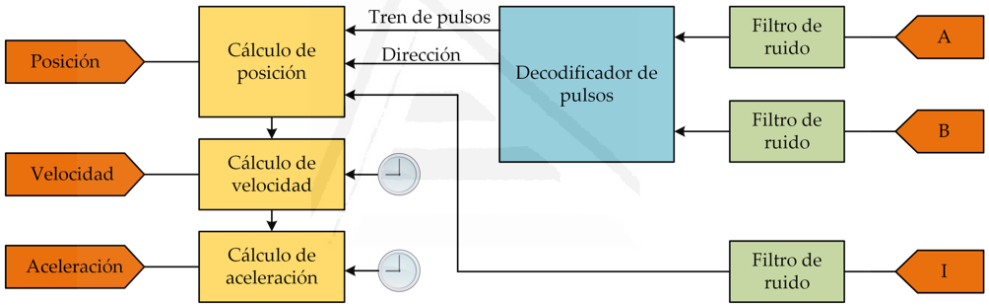


Figura 3.15 Esquema funcional del submódulo Interfaz Encoder

de transmisión entre el motor y la articulación. La velocidad se calcula como el cambio de posición en una unidad del tiempo. Y, por último, la aceleración es el cambio de velocidad en una unidad del tiempo.

3.3.1.2.2 Interfaz del robot Mitsubishi PA-10

Como se ha comentado anteriormente, el robot Mitsubishi PA-10 emplea el protocolo de comunicación ARCNET para enviar comandos al Servo Driver, y recibir datos sobre el estado de sus articulaciones. El comando de control para el PA-10 es una trama de 256 bytes, donde en los primeros 2 bytes de define el ID del transmisor y receptor (ver la Figura 3.16). Y los últimos 44 bytes tienen, entre otros parámetros, los valores de par/velocidad que queremos que se apliquen a los motores del robot.

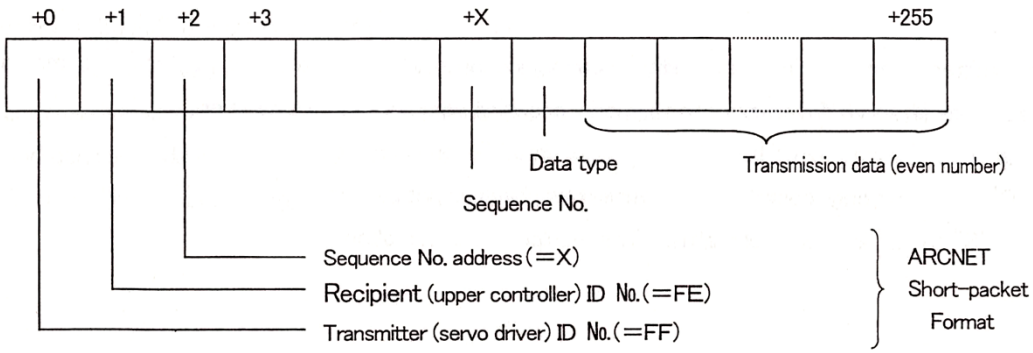


Figura 3.16 El formato del paquete en el protocolo ARCNET usada para el PA-10 (Fuente: [Mitsubishi-2005])

La Figura 3.17 muestra la organización de los 44 bytes de datos en el paquete ARCNET. En un comando de control se reservan 6 bytes para cada eje (42 bytes en total para los 7 ejes del PA-10), y los últimos 2 bytes son para la CPU de comunicación del Servo Driver. Por ejemplo, para aplicar un par determinado en el eje S1, es necesario poner el bit 2 del primer byte (*torque, velocity switching torque/velocity*) a 1, para elegir que se desea hacer control del motor por par, y luego enviar el valor del par en los dos bytes reservados para este valor (*Torque command value).* Cuando el Servo Driver recibe un comando de control, tiene que responder enviando el estado de los ejes del robot en una trama similar a la de la Figura 3.16, pero los bytes de datos en la respuesta son 72 bytes y tienen el formato de la figura Figura 3.17. En la trama de respuesta hay 10 bytes reservados para cada eje que contienen información real de los parámetros de cada eje, como la posición angular, la velocidad actual, y el par actual. Los primeros dos bytes (*Servo CPU*) sirven como bits de testigo (*flags*) que reflejan el estado de toda la electrónica del controlador y el robot en general, como los finales de carrera, por ejemplo, y permiten también diagnosticar y descubrir anomalías en el funcionamiento del Servo Driver (para más detalles consultar el manual).

3.3.2 Módulos independientes del hardware

3.3.2.1 Algoritmos de Visión

El módulo de Algoritmos de visión es un módulo genérico e independiente de la cámara usada. Este módulo recibe todos los parámetros de la imagen (Formato,

resolución, etc.) del módulo Digitalizador de Video, el cual es un módulo específico e implementado a medida para la cámara empleada (ver Sección 3.3.1.1). Este módulo recibe los parámetros de la imagen sólo una vez, al poner sistema en marcha, y a continuación recibe los datos de la imagen en formato único: el píxel y sus coordenadas espaciales (x, y) en la imagen.

La tarea que cumple este módulo es procesar la imagen para segmentar los objetos del patrón utilizado en la tarea de control visual y, finalmente, calcular los centros de masas de estos objetos, y enviarlos al módulo Ley de Control. En el Capítulo 4 se explica la implementación de este módulo en la FPGA, dónde se propone un diseño de los

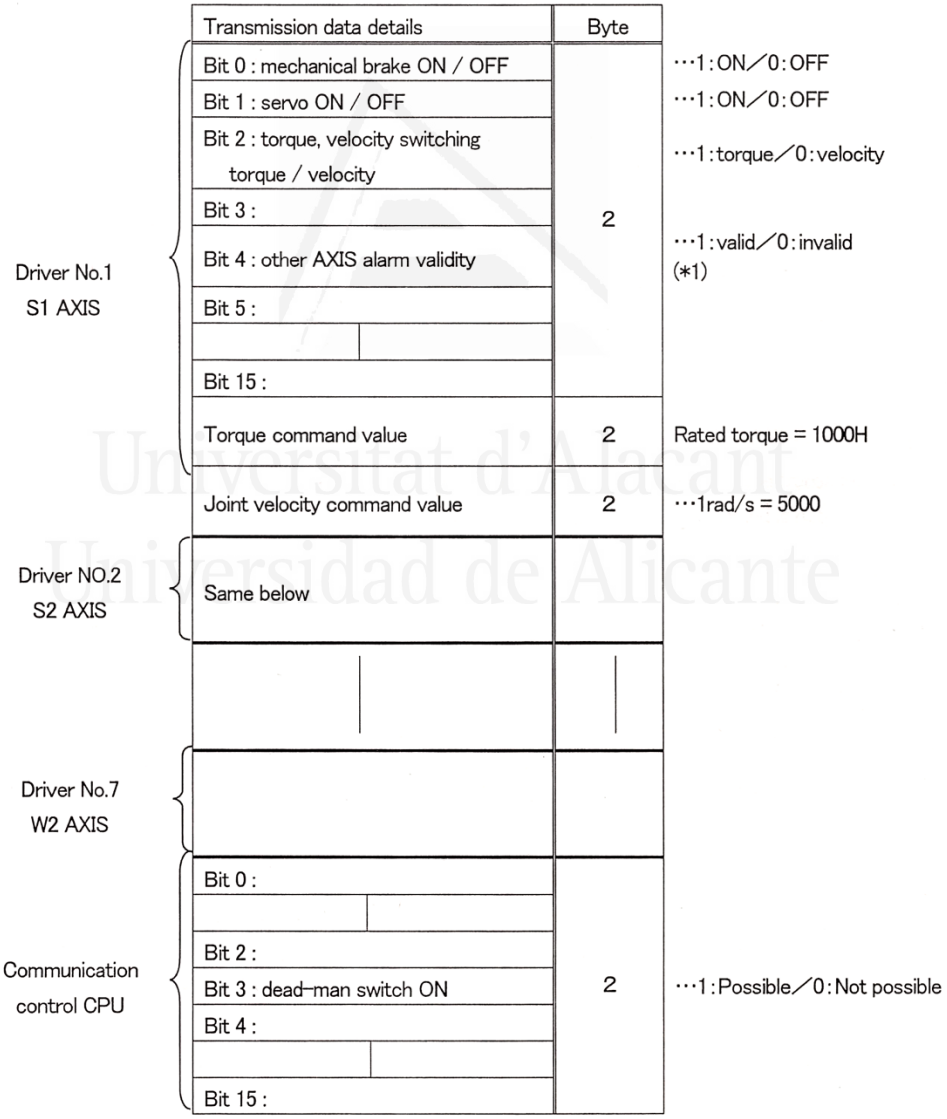


Figura 3.17 La trama de datos del comando de control del PA-10

	Transmission data details	Byte	
Driver No.1 S1 AXIS	Servo CPU (*1)	2	
	Joint angle	4	···360[deg] = 4000H × 50
	Current velocity	2	
	Current torque	2	
Driver No.2 S2 AXIS	Same below		
Driver No.7 W2 AXIS			
Communication control CPU	Communication control CPU (*2)	2	

Figura 3.18 La trama de datos de la respuesta del PA-10

algoritmos de visión de forma que se ejecutan en paralelo usando la técnica cauce segmentado, una técnica con la que se garantiza la optimización temporal y espacial de los recursos de una FPGA.

3.3.2.2 Ley de Control

Los algoritmos de control visual vienen implementados en este módulo, que es un módulo genérico y no depende directamente del robot empleado. Este módulo debe recibir todos los parámetros del robot una vez puesto en funcionamiento el módulo especifico Interfaz de los Robots (ver Figura 3.12) como, por ejemplo, el número de grados de libertad del robot, sus parámetros dinámicos (masas, inercias, dimensiones y configuración de eslabones, etc.), etc.

El módulo Ley de Control recibe dos tipos de datos: las características visuales (los centroides en el plano imagen de los objetos del patrón) del módulo Algoritmos de Visión, y los datos del robot (Posición, velocidad y aceleración angular de cada eje) del módulo específico Interfaz de los Robots. La salida de este módulo es un vector de pares que hay que aplicar a los motores del robot para anular el error en el espacio imagen.

En el capítulo 5 se proponen varias estrategias y algoritmos para conseguir realizar tareas de posicionamiento e incluso seguimiento de trayectorias, y se muestra su

implementación paralela en la FPGA. En este capítulo, los algoritmos propuestos en esta Tesis se verifican en robots reales, obteniendo respuestas en tiempo real y comprobando su rendimiento frente a otras implementaciones.

3.3.3 Otros módulos implementados en la FPGA

A parte de los módulos que se han explicado hasta ahora, y que forman el núcleo del software del sistema de control, hay otros módulos que se han integrado en la FPGA como herramientas que permitan configurar o monitorizar los parámetros y variables del sistema. Entre los módulos implementados de este tipo se pueden destacar los siguientes:

1. Módulo de interfaz HDMI. Se trata de una interfaz software que permite visualizar la imagen de salida de cualquier fase del procesamiento de imagen en un monitor conectado al puerto HDMI de la KC705. Esto permite ver la imagen capturada para verificar el correcto funcionamiento, o las condiciones de iluminación. También permite elegir un umbral adecuado, mostrando la imagen en la fase de binarización, o ver los objetos detectados por el algoritmo de detección de objetos e incluso ver sus centroides en la misma pantalla.
2. Módulo de configuración de la cámara. El estándar *CameraLink* tiene integrados en su protocolo dos hilos para comunicación serie, Rx y Tx. En el caso de la cámara empleada en los experimentos mostrados en los distintos capítulos de esta Tesis, esta comunicación permite configurar los parámetros de la cámara (fps, tiempo de exposición, etc.). El fabricante de la cámara proporciona un software para Windows que permite configurar dichos parámetros. Se ha implementado un módulo que funciona con un adaptador de comunicación entre la cámara (conectada a la FPGA en el FMC LPC) y el PC (conectado a la FPGA por puerto serie).
3. Módulo Interfaz del Terminal de Control. Es un módulo cuya función es gestionar el Terminal de Control explicado anteriormente en la Sección 3.2.2.3. Se ha implementado el código VHDL para gobernar la pantalla LCD, y atender las acciones recibidas a través de los elementos de este terminal.

3.3.4 Interfaz en el PC

Para facilitar la interacción con el software implementado en la FPGA, así como para poder recibir/enviar datos de/a la FPGA en línea, se ha desarrollado un software Interfaz para Windows utilizando el lenguaje de programación C#. Este software permite el intercambio de datos entre la FPGA y el PC por puerto serie. La Figura 3.19 muestra las diferentes funciones del entorno creado, que incluye:

1. Ventana Señales. Esta ventana representa la evolución de las distintas señales del sistema en tiempo real y las muestra también en modo fuera de línea (cargar experimentos anteriormente almacenados en archivos). En la parte inferior de esta ventana se pueden seleccionar las señales que interesa visualizar en cada instante, así como su color.
2. Ventana Control. Permite elegir el modo de control: Manual o Automático, y en el caso de elegir Automático, también es posible elegir uno de los controladores implementados en la FPGA. El botón Ajustes abre una ventana que permite cambiar los datos dinámicos, o cambiar las ganancias de los controladores en tiempo real sin la necesidad de retocar el código VHDL.
3. Ventana Espacios. Muestra gráficamente el espacio Cartesiano del robot y el espacio imagen, donde se puede ver la postura real del robot. También puede verse la posición inicial y final del extremo del robot y de las características visuales en el espacio robot e imagen, respectivamente. Igualmente, se puede mostrar gráficamente la trayectoria seguida por el extremo del robot o las características visuales en una tarea de posicionamiento o seguimiento de trayectorias de control visual.
4. Ventana Archivo. Se trata de un conjunto de botones que generan y manejan los archivos de los experimentos. Como, por ejemplo, guardar un experimento (se guardan todos los datos en un archivo de texto en una carpeta en el disco duro del PC), cargar un experimento guardado y explorar todos sus detalles, o generar gráficas .fig directamente en Matlab de las variables importantes, como el par o el error en imagen.

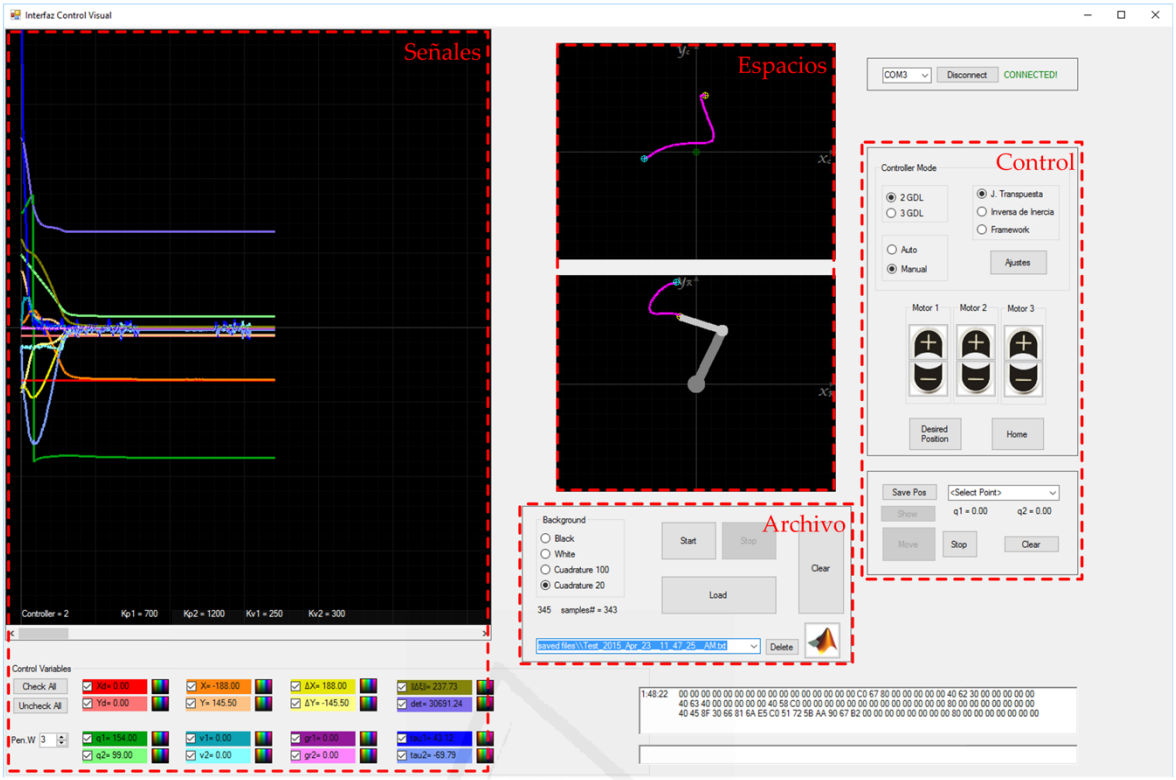


Figura 3.19 Funciones del interfaz desarrollado

3.4 Conclusiones

La principal ventaja de un sistema de control visual dinámico basado en imagen es que se controlan directamente los pares que permiten posicionar al robot mediante visión. Estos sistemas requieren un conocimiento exhaustivo de los parámetros dinámicos del robot a controlar, pero las ventajas que proporciona la eliminación del bucle de control interno de los robots son muchas. Teniendo en cuenta la dinámica del robot, se alcanza el objetivo de una manera más rápida y precisa. Sin embargo, prácticamente ningún fabricante de robots manipuladores proporciona acceso a su controlador interno, dejando en el control visual cinemático, o indirecto, la única posibilidad de guiado mediante visión. Para conseguir que el esfuerzo realizado en el diseño de un controlador directo sea fructífero, se requiere una alta velocidad de proceso, tanto en el cálculo de las posiciones de las características visuales en cada momento, como de la ley de control una vez se dispone de esta información. Las FPGAs proporcionan herramientas muy interesantes para el desarrollo de sistemas de visión artificial que, no sólo permiten mejorar

enormemente los tiempos de proceso, sino que además proporcionan respuestas en un tiempo acotado, como se podrá ver en el Capítulo 4.

En este Capítulo se ha descrito la arquitectura hardware que permite realizar pruebas de nuevos controladores visuales dinámicos sobre dos robots. En realidad, la arquitectura propuesta pretende ser una arquitectura semiabierta. Para ello, se ha propuesto un sistema modular, en el que se puedan separar claramente los módulos que son dependientes del hardware, o específicos, de los que no lo son. Así, atendiendo al robot manipulador que se desea controlar, se describen módulos dependientes del hardware que permiten obtener información de la posición, velocidad y aceleración articulares actuales. También se proporciona un módulo que permite enviar pares al robot. A partir de estos módulos, que evidentemente se tendrían que programar para cada nuevo robot que se pretenda controlar, se hace uso del módulo de Control, que ya es independiente del hardware. Este módulo de control se detalla en el Capítulo 5.

Además del robot, la FPGA necesita obtener información de la cámara. Para ello, de nuevo, se diseña un módulo dependiente del hardware que permite comunicarse en este caso con una cámara rápida industrial con interfaz *CameraLink*. A partir de aquí, el módulo de Algortimos de visión (que se describe con gran detalle en el Capítulo 4), es informado de información como el tipo de imagen (color o escala de grises, por ejemplo), sus dimensiones, así como cualquier particularidad del tipo de cámara.

Toda la electrónica necesaria para la comunicación, tanto con la cámara, como con los dos robots utilizados en los experimentos mostrados en esta Tesis, se ha descrito de manera pormenorizada. Estos componentes electrónicos se diseñan de manera específica para el robot que se desea utilizar y la cámara que lo guiará. Son, por tanto, los componentes dependientes del hardware de la arquitectura propuesta.

Además de la comunicación con la cámara o con el robot, se han desarrollado otros módulos que permiten interactuar con el sistema en tiempo de ejecución. Se ha implementado un controlador HDMI para poder ver el procesamiento de la imagen en las distintas etapas. También se ha descrito un interfaz programado en C# bajo Windows, que permite interactuar en línea con la FPGA. De esta forma es posible no sólo recibir

información de la FPGA, sino también enviarle nuevos valores de los parámetros del sistema.

4 Sistema de visión para control visual basado en FPGA

El objetivo del módulo de visión en los sistemas de control visual es proporcionar información visual procesada de manera continua al controlador. Desafortunadamente, el procesamiento del flujo de imágenes es una tarea compleja, con alta demanda de cómputo, sobre todo para las frecuencias de captura cada vez mayores que pueden proporcionar las cámaras. El uso de los dispositivos FPGA permite la ejecución de tareas de procesamiento con un alto refresco de imágenes en tiempo real. Gracias a su paralelismo, el procesamiento puede hacerse en cauces segmentados para conseguir bajas latencias y altas prestaciones. Además, el bajo consumo y el tamaño compacto de estos dispositivos FPGA son aspectos claramente a tener en cuenta para su uso como plataforma en la que implementar sistemas de visión avanzados.

En este Capítulo, se propone una arquitectura para el proceso de imágenes en tiempo real. Esta arquitectura (como se describe en el Capítulo 3) se ha diseñado para ser parcialmente abierta, para lo que se divide en dos partes funcionales: el digitalizador de vídeo (que es dependiente del hardware) y el módulo de visión (que no depende del hardware). Se detallará cada fase del proceso de un fotograma para obtener los centroides de los objetos detectados. La arquitectura propuesta se ha implementado en una FPGA y se ha analizado con pruebas reales utilizando una cámara rápida que opera hasta a 340 fps. Finalmente, se realiza una comparativa entre la implementación sobre FPGA propuesta y una implementación sobre PC tradicional.

4.1 Introducción

Los sistemas de visión por computador están basados tradicionalmente en una arquitectura secuencial. Así, los distintos procesamientos de una imagen se ejecutan uno

detrás de otro en una sucesión. El programa se divide en una secuencia de operaciones aritméticas y lógicas que se ejecutan en la ALU (del inglés *Arithmetic Logic Unit*). El resto de la CPU se diseña para proporcionar a la ALU los datos necesarios. El algoritmo se compila en una secuencia de instrucciones, que se usan para controlar las operaciones que se ejecutarán en la CPU y la ALU en cada ciclo de reloj. De esta forma, la operación básica que realiza la CPU es buscar una instrucción en la memoria, decodificarla para determinar qué operación se debe realizar, y ejecutarla.

En contraste, una aproximación paralela implementa cualquier instrucción de un algoritmo determinado en procesadores separados. Sin embargo, si el algoritmo es predominantemente secuencial, donde cada una de sus instrucciones dependen de los datos obtenidos en instrucciones previas, la ganancia que se obtendría con una aproximación paralela sería prácticamente nula. Con el objetivo de obtener una implementación paralela útil, el algoritmo debe ser susceptible de ser separado en partes independientes. Para alcanzar cualquier mejora importante, la porción de algoritmo que puede ser implementado de forma paralela debe ser significativa. Afortunadamente, el procesamiento de imagen es inherentemente paralelo, especialmente en las tareas de bajo y medio nivel.

Se pueden aplicar dos conceptos de paralelismo distintos a los algoritmos de visión: paralelismo temporal y paralelismo espacial. Los algoritmos de procesamiento de imagen implementan, generalmente, una secuencia de operaciones de imagen. Asignando cada procedimiento a diferentes procesadores, el procesamiento de imagen puede alcanzar una forma de paralelismo temporal. Este paralelismo temporal puede verse como una arquitectura basada en cauce segmentado, donde cada procesador ejecuta una operación sobre los datos y manda su resultado a la siguiente entidad. Habitualmente, los algoritmos de procesamiento de imagen contienen uno o varios bucles. Estos bucles iteran sobre los píxeles de la imagen, aplicando la misma operación, independientemente de su valor. Esta clase de paralelismo se denomina paralelismo espacial. Para aprovechar el paralelismo espacial, la imagen se debe particionar de alguna manera (ver Figura 4.1). Después de esto, un procesador diferente puede hacerse cargo de cada parte de la imagen.

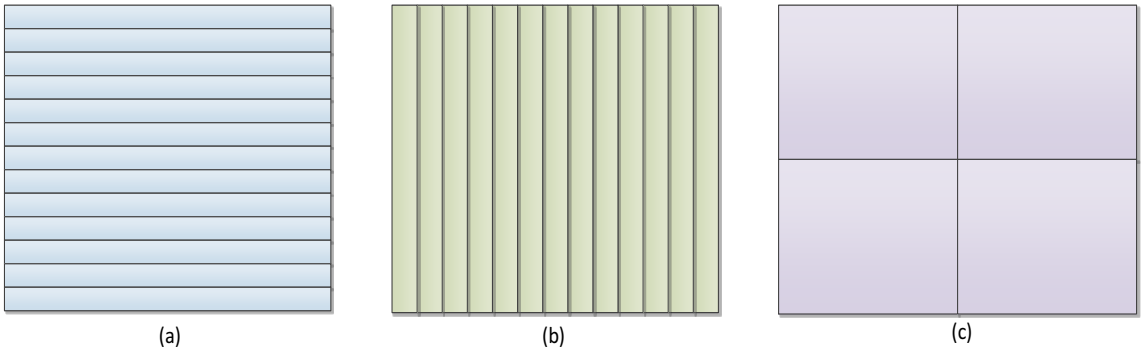


Figura 4.1 (a) Particionado de fila. **(b)** Particionado de columna. **(c)** Particionado de bloque

Uno de los cuellos de botella más comunes en el procesamiento de imagen es el tiempo necesario (y ancho de banda) para leer la imagen de la memoria y almacenarla después de ser procesada. Convertir el paralelismo espacial en un paralelismo temporal puede ayudar a minimizar este problema. Para conseguirlo, la imagen es transmitida, es decir, se lee y escribe secuencialmente como un vector de bytes, habitualmente a una velocidad de un píxel por ciclo de reloj (ver Figura 4.2). A partir de esta imagen en forma de vector, el tiempo empleado en el procesamiento de la imagen se obtiene de la cantidad de tiempo necesario para leer/escribir la imagen y la latencia de procesamiento. En la mayoría de operaciones, la latencia normalmente lleva mucho menos tiempo que la lectura misma de toda la imagen. Así que, si se consigue implementar el algoritmo de procesamiento de imagen en un flujo, el tiempo de respuesta estará dominado por la frecuencia a la que se obtienen las imágenes.

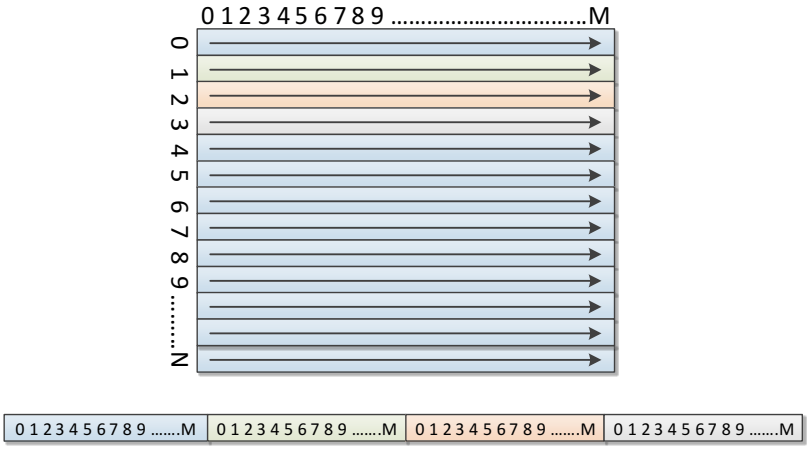


Figura 4.2 Vector de datos de la imagen

A la hora de describir la tarea que debe desarrollar un sistema de visión artificial, se encuentra bien establecida en la literatura la división en tres niveles de abstracción: tareas de bajo nivel, tareas de medio nivel y tareas de alto nivel [Adelson-1994]. Las tareas de visión de bajo nivel consisten en operaciones a nivel de píxel como filtrado, estimación de intensidad, segmentación y detección de bordes. En una tarea de bajo nivel, el sistema de visión artificial tiene que lidiar normalmente con una gran cantidad de datos. Estas tareas consisten en operaciones en pequeños entornos de un píxel, como la segmentación, el filtrado o las operaciones básicas de punto. Sin embargo, las tareas de bajo nivel requieren generalmente operaciones relativamente simples (como multiplicaciones y sumas). La visión de medio nivel consiste en operaciones de agrupamientos de píxeles, como la determinación de características de objetos o el etiquetado de regiones. Estas tareas se caracterizan de nuevo por acceso a datos locales, pero esta vez con operaciones a nivel de píxel más complejas. Finalmente, las tareas de visión de alto nivel están más orientadas a la toma de decisiones, como el reconocimiento de objetos, de caras o escenas. Estas tareas engloban acceso a datos remotos y algoritmos no deterministas y complejos.

A continuación, se muestra un resumen de los trabajos de visión y FPGA que se encuentran en la literatura. Estos trabajos se han dividido en tres categorías siguiendo las siguientes reglas: si el objetivo principal de la tarea es una mejora de la imagen, la tarea se catalogará como de bajo nivel; las tareas que operan en los píxeles para producir características en la imagen serán consideradas como de medio nivel; y finalmente, la fase de toma de decisiones se clasificará como de alto nivel. En el último subapartado se describirán los trabajos más relacionados con la propuesta de sistema de visión artificial embebido para la detección de objetos y el cálculo de su centro de masas.

4.1.1 Tareas de visión de bajo nivel

Una de las tareas de visión de bajo nivel es una convolución. Esta fue, además, una de las primeras tareas de procesamiento de imágenes implementadas en una FPGA [Bravo-2006]. En este trabajo, las imágenes proporcionadas por un sensor de alta resolución se enviaban a la FPGA para, entonces, aplicar una convolución con una máscara sobre la imagen y pasar el resultado de la operación a un PC.

En [Pujari-2014] se describe la implementación sobre una FPGA de los filtros de Sobel y Prewitt, que permiten la detección de bordes en vídeo y aplicaciones de procesamiento de imagen. Usando la aproximación basada en modelo (del inglés *Model Based Approach*), se describen técnicas de procesamiento de imágenes para la detección de bordes en secuencias de vídeo obtenidas directamente de una webcam, así como de la lectura directa de un vídeo desde el disco duro. Se utiliza el kit de FPGA DE2 de ALTERA, que dispone de una FPGA CYCLONE II para implementar completamente el algoritmo de detección de bordes en un vídeo. El vídeo, o la imagen de entrada, viene de una cámara NTSC/PAL, y las imágenes, ya con los bordes detectados, se muestran en un monitor VGA. Se utiliza el software de ALTERA QUARTUS-II para compilar el código VHDL con la implementación funcional de todos los procesos.

En [Hentati-2014] se propone una implementación hardware de operaciones morfológicas basadas en una técnica de reconfiguración dinámica y parcial (DPR). Esta técnica permite reconfigurar una parte de la FPGA con diferentes funcionalidades en tiempo de ejecución. Es una solución prometedora para incrementar el rendimiento del sistema. El diseño permite al usuario elegir la operación morfológica adecuada (erosión o dilatación) de acuerdo con las restricciones de la imagen de entrada. Esta implementación se realiza sobre una Virtex-5 de Xilinx, de la que se muestra la ocupación y el tiempo invertido en la reconfiguración.

La estimación de movimiento es una técnica de procesamiento de imágenes compleja, que puede utilizarse para aplicaciones tales como interpolación temporal de secuencias de imágenes, compresión de vídeo, segmentación de movimiento, o seguimiento [Torres-2014]. Los algoritmos de flujo óptico han sido ampliamente utilizados para la estimación de movimiento. En [Botella-2011] Botella et al. presentan un trabajo desarrollado sobre un dispositivo FPGA de Xilinx que realiza dos tareas de visión de bajo nivel: estimación de flujo óptico de la familia de gradientes y momentos ortogonales variantes. Estos dos bloques se utilizan para una tarea de nivel medio (seguimiento). El sistema descrito en ([Díaz-2004], [Díaz-2006]) muestra cómo un circuito óptico de estimación de flujo puede ser implementado utilizando una plataforma FPGA para lograr la computación en tiempo real. La diferencia en esta propuesta radica en el hecho de que los autores implementan

un modelo de gradiente de Lucas y Kanade clásico [Lucas-1981], comparando diferentes métodos de estimación de flujo óptico para evaluar el rendimiento del sistema.

Los sensores de imagen de fóvea adaptativos definen estructuras reconfigurables no concéntricas para campos de visión rectangulares. Siguiendo procedimientos utilizados en pirámides de visión, a partir de imágenes de resolución uniforme proporcionadas por una cámara, los niveles superiores se computan progresivamente, reduciendo la resolución y el volumen de datos. En [Camacho-2000], [Coslado-2004] los sensores de imagen de fóvea adaptativos se implementan en una FPGA. Cada píxel de la imagen de resolución completa es promediado en una tarea de visión de bajo nivel. Otra aplicación de procesamiento de imagen interesante donde una FPGA mejora el rendimiento es en la coincidencia de huellas dactilares en línea. En [Fons-2006] la implementación sobre una FPGA del algoritmo de análisis estructural consiste en un bloque de máquina de estados finitos responsable de gestionar el análisis del entorno del píxel. Con el objetivo de acelerar el cálculo de las distancias y ángulos entre los puntos de minutiae, se implementa un coprocesador CORDIC. El procesador CORDIC se usa habitualmente cuando no hay disponible ningún multiplicador, ya que las únicas operaciones que requiere son la suma, la resta, el desplazamiento de bit y las consultas de tablas [Volder-1959].

La correspondencia estéreo es una tarea de visión de bajo nivel. No siempre se ha considerado dentro de este grupo. Sin embargo, ninguna información de alto nivel ayuda al observador a emparejar puntos obtenidos de dos imágenes de un par estéreo. Los algoritmos de emparejamiento estéreo pueden ser clasificados en dos aproximaciones (global y local) basadas en las estrategias usadas para la estimación. Las aproximaciones globales obtienen resultados más precisos, pero incrementando el coste computacional. En [Barranco-2012], Barranco et al. implementan dos alternativas distintas para calcular el vector de disparidad de un sistema de visión activa: una técnica basada en gradiente (el algoritmo local de Lucas y Kanade), y un algoritmo basado en fase detallado en [Sabatini-2010] (también un algoritmo local). La primera técnica estima pequeñas disparidades locales, asumiendo la persistencia de la intensidad o brillo de un píxel entre las imágenes izquierda y derecha, mientras que la segunda técnica calcula la disparidad usando la información de fase para diferentes orientaciones, de una manera independiente, por lo

tanto, del contraste. Ambos métodos se han implementado en una Virtex4 XC4vfx100 de Xilinx, alcanzando una frecuencia de trabajo de 32 fps, con una resolución de 640x480 píxeles. Los experimentos que muestran validan su propuesta y concluyen que la mejor opción de las probadas es la del algoritmo de Lucas y Kanade. Gil et al. [Gil-2004] describen una aplicación de guiado de un robot móvil, basada en una FPGA, que calcula la correspondencia estéreo de dos imágenes provenientes de un par estéreo. El algoritmo estéreo implementado está basado en la transformada Census, descrita en [Zabih-1994]. En [Magdaleno-2010], [PérezJM-2009b], uno de los algoritmos globales más utilizados se implementa en una FPGA: el algoritmo de propagación de creencias (del inglés *belief propagation*). Es un algoritmo de emparejamiento de puntos de gran precisión, pero requiere una gran cantidad de memoria. Este requisito de memoria es incluso mayor para imágenes de alta definición. La arquitectura propuesta utiliza una Virtex 5 330 VLX de Xilinx para reducir el tiempo de ejecución necesario para obtener mapas de profundidad de alta definición.

Es muy común encontrar en la literatura trabajos que implementan sólo una parte del sistema sobre una FPGA y el resto es implementado en un PC o cualquier otro dispositivo. Normalmente, el procesamiento de imágenes se implementa embebido en una FPGA, mientras que las tareas de decisión (correspondientes con tareas de visión de alto nivel) se desarrollan sobre procesadores convencionales. En [Lorenz-2013] la placa FPGA permite operaciones concurrentes ejecutadas simultáneamente, como la adquisición de la imagen y el procesamiento de la información visual. El bloque de procesamiento incluye tareas como el filtrado de imagen, ajuste de la umbralización y detección de bordes. La imagen procesada con este bloque es posteriormente enviada a un ordenador para medir el nivel de líquido en una aplicación de destilación de membrana.

El módulo de visión propuesto en esta Tesis aplica varios algoritmos de procesamiento de imagen de bajo nivel al vector de la imagen que llega desde el digitalizador de vídeo para proporcionar al bloque de detección de objetos una imagen adecuada. Como se describe más en detalle en el Apartado 4.2, la cámara envía dos píxeles cada ciclo de reloj de píxel a una frecuencia de 80 MHz, y una frecuencia de refresco de 340 fps. Trabajando a esta velocidad, la imagen muestra cierto tipo de ruido que necesita filtrarse. El bloque de

preprocesamiento propuesto en esta Tesis redimensiona la imagen fusionando cada dos píxeles recibidos junto con los dos píxeles vecinos inferiores, es decir, este bloque calcula la media de cada 4 píxeles y envía el resultado al siguiente bloque. Como resultado, las dimensiones de la imagen se reducen a la mitad y el hecho de promediar cada 4 píxeles hace que este bloque se comporte, además, como un filtro de media, ayudando así a reducir el nivel de ruido de la imagen. Posteriormente, se aplica una operación de umbralizado y, a continuación, una fase opcional de erosión para terminar de eliminar ruido en la imagen. La umbralización es la tarea de visión de bajo nivel más básica, ya que su cálculo depende únicamente del valor de intensidad del píxel actual, mientras que la operación de erosión requiere algo más de información sobre los píxeles vecinos para calcular el valor de salida de un píxel, para lo que se necesita el uso de memorias intermedias que proporcionen al menos las dos últimas líneas de la imagen.

La Figura 4.3 muestra un esquema de una tarea de bajo a medio nivel implementada sobre una FPGA. Las posibilidades de paralelismo que ofrecen las FPGAs tienen efectos en la construcción del sistema de visión. La implementación de una arquitectura en cauce segmentado en una FPGA permite operar a la misma frecuencia que se proporcionan los píxeles. Dado que el consumo de energía está directamente relacionado con la frecuencia de reloj, una menor frecuencia implica una menor demanda de energía por parte del sistema. La tarea de visión descrita en la Figura 4.3 es una aproximación FPGA típica a una tarea de procesamiento de imágenes.

Normalmente, los datos de la imagen se transmiten en serie, lo que encaja perfectamente en una implementación hardware, especialmente si es posible interactuar directamente con la cámara. De todos modos, un bloque (representado en la Figura 4.3 como una interfaz de E/S conectada directamente a la cámara) realiza la comunicación con la cámara para recibir los píxeles que fluyen desde el sensor. Este bloque es el responsable de implementar el protocolo requerido (I2C, CameraLink, etc.) para comunicarse con el dispositivo de captura de datos. El flujo de píxeles es enviado al bloque de procesamiento básico de imágenes (bloque azul de operaciones de punto en la Figura 4.3).

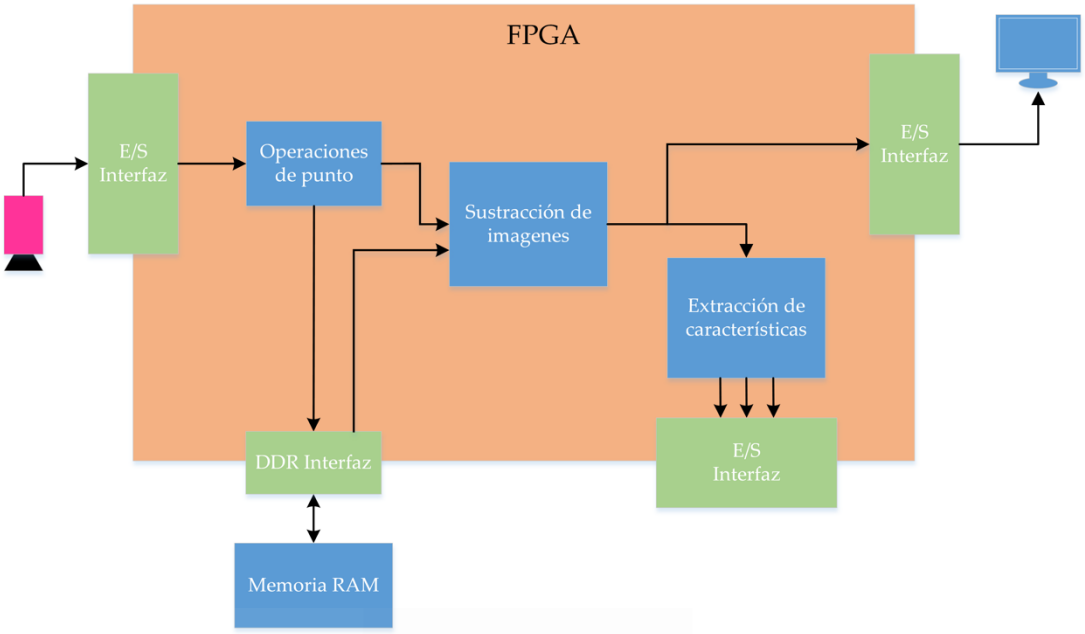


Figura 4.3 Esquema de un sistema de visión artificial embebido en una FPGA. Seguimiento de cuatro puntos y cálculo de su centro de masas a la frecuencia de muestreo de la cámara

Las operaciones de punto se han utilizado ampliamente en términos de mejora de contraste, segmentación, filtrado de color, detección de cambios, enmascaramiento y muchas otras aplicaciones del procesamiento de imagen. Estas operaciones contienen la peculiaridad de que el píxel de salida depende únicamente del valor del píxel de la entrada. La Figura 4.4 representa un ejemplo de este tipo de módulo, en el que se realiza una operación de mejora de contraste simple a la imagen de entrada. Las constantes *a* y *b*, junto con dos operaciones matemáticas simples sobre el valor del píxel de entrada proporcionan un nuevo valor de luminancia. Este valor puede exceder el rango de valores representables. Por lo tanto, el resultado debe ser recortado. En la Figura 4.4 se puede ver esta operación de recorte sobre el valor de salida. Operar con el valor de entrada puede mejorar el rendimiento en un esquema paralelo porque tanto las operaciones matemáticas como las comparaciones lógicas pueden procesarse simultáneamente con dos procesadores. El resultado de este módulo se puede almacenar en algún tipo de memoria del dispositivo (RAM DDR2 en la Figura 4.3). Este último paso no es estrictamente necesario. Sólo se requiere un almacenamiento de memorias intermedias para la sincronización del sistema.

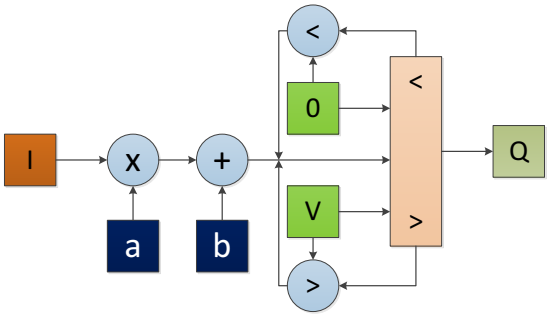


Figura 4.4 Operación de punto básica en una FPGA. La luminancia de un píxel de entrada se multiplica por una constante *a*, y posteriormente se suma el resultado con la constante *b*. El valor obtenido se recorta antes de almacenar el nuevo valor Q del píxel

Las operaciones de punto son sólo las tareas básicas de visión de bajo nivel. Normalmente, a partir de la imagen mejorada que se obtiene de un módulo de operación de punto, el sistema de visión artificial realiza otras operaciones de bajo nivel como, por ejemplo, un filtro de media de imagen. Los filtros o las operaciones de seguimiento de los objetos visuales tienen en común que necesitan más información además del valor del píxel que se está procesando. Para ello, es esencial disponer de la arquitectura necesaria para obtener dicha información (estructuras vectoriales, memorias intermedias, etc.). La Figura 4.5 muestra algunas iteraciones de un filtro de media calculado en una FPGA. A la izquierda se representa la imagen de entrada para cada iteración. En rojo se enmarca la máscara de convolución empleada para calcular el nuevo valor de intensidad del punto central en la iteración correspondiente, mientras que las memorias intermedias con la información de filas completas necesarias para la operación se representan en un azul más oscuro y verde. A la derecha del esquema se muestran también los valores de las memorias intermedias de las filas para cada iteración. Las memorias intermedias con la ventana de la máscara se representan en naranja. Las memorias intermedias con las filas y la ventana se actualizan iterando sobre el vector con la información de la imagen. El paralelismo se aprovecha gracias a estas memorias intermedias. A partir de la memoria intermedia de la ventana se puede calcular la media del píxel y sus vecinos en cada iteración. El resultado es un valor de píxel válido de la imagen filtrada de salida.

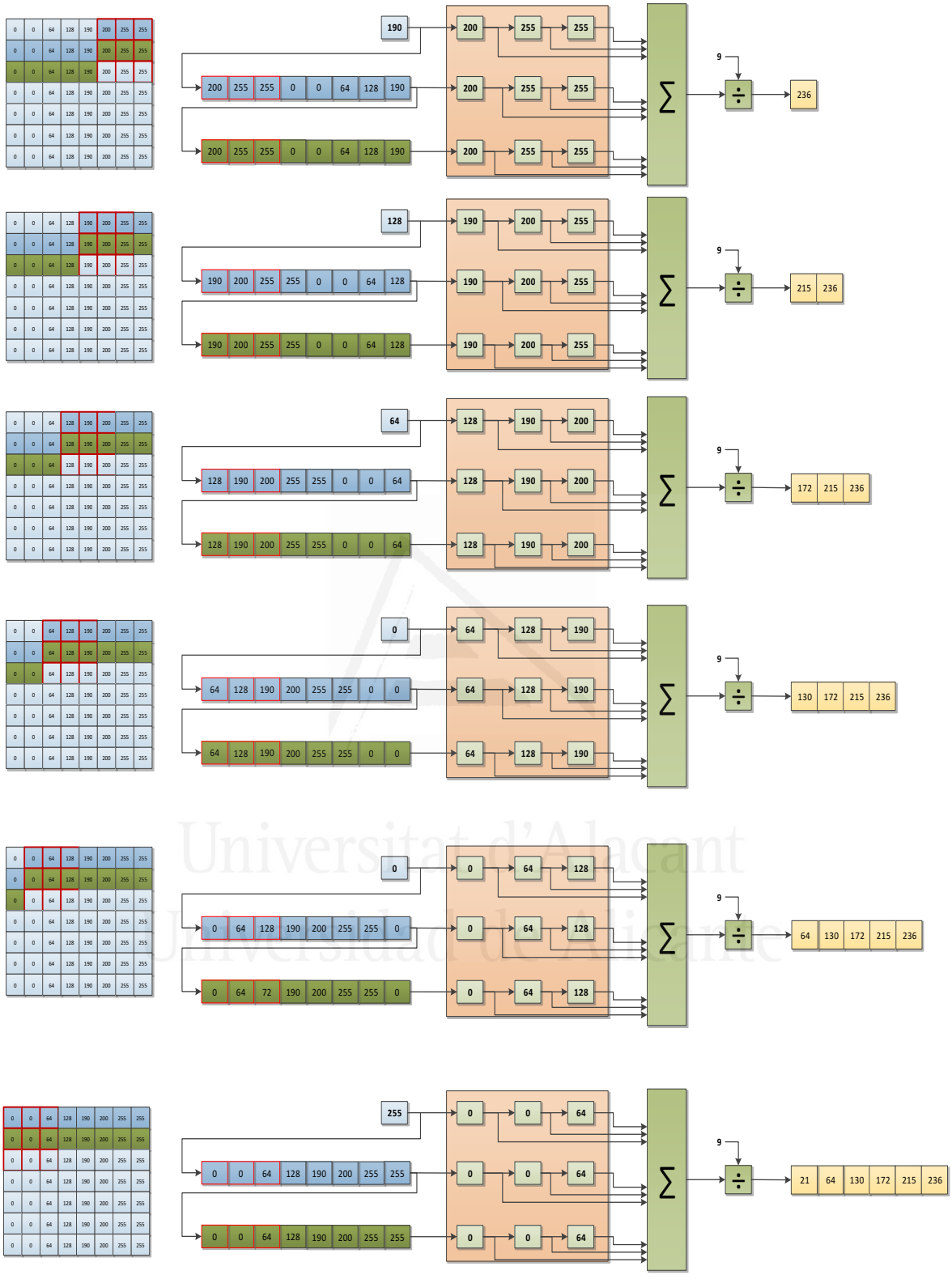


Figura 4.5 Filtro de media de una imagen en una FPGA. Diferentes iteraciones sucesivas de la operación de filtrado

4.1.2 Tareas de visión de medio nivel

Normalmente, la entrada de un algoritmo de medio nivel es una imagen procesada en una tarea de bajo nivel. La información proporcionada en esta fase se corresponde con características de la misma imagen o de los objetos contenidos en la imagen. Algunos ejemplos serían la estimación de la posición de los objetos segmentados en la imagen, la magnificación, orientación, detección de esquinas o bordes, o etiquetado de regiones.

La gran mayoría de trabajos que se implementan en una FPGA buscan obtener una cámara más inteligente. En [Albo-Canals-2012] se propone un prototipo de cámara binaria digital de redes neuronales no lineales (del inglés Cellular Nonlinear Network (CNN)) basado en una FPGA IGLOO de Actel. La cámara se utiliza para guiar un robot Mindstorm de LEGO. El sensor propuesto debe ajustarse a los requerimientos de sensores electrónicos del Mindstorm de LEGO, no sólo en términos de tamaño del dispositivo, sino también en cuanto a consumo (menos de 140 mA). El bajo consumo de la FPGA y su reconfigurabilidad se explotan en este trabajo para procesar una detección de esquinas. El trabajo presentado en [Batlle-2002] describe el uso de una FPGA para calcular la posición y orientación relativas de un robot submarino con referencia a una tubería bajo el mar. Después de una binarización de la imagen, la FPGA calcula la distancia entre las líneas de la tubería que se ven en la imagen binarizada, la posición del centro de la primera línea y detecta si una de las líneas ha desaparecido de la imagen. A partir de esta información, un DSP calcula parámetros como la orientación y la distancia entre el robot y la tubería. Los procesamientos de bajo y medio nivel locales acaban aquí y esta información se envía al equipo que se encarga en última instancia de las últimas etapas relativas a tareas de interpretación y entendimiento de la escena (tareas de visión de alto nivel). Una de las tareas más comunes realizadas por la visión artificial es la detección de objetos moviéndose sobre un fondo estático. La navegación, el seguimiento o las aplicaciones de vigilancia están directamente relacionadas con el análisis de movimiento. En [Bravo-2007], [Bravo-2010] se implementa en una FPGA el análisis de componentes principales (del inglés Principal Component Analysis (PCA)) para la detección de objetos móviles en una escena. En estos trabajos se presenta el primer desarrollo completo del algoritmo de PCA sobre una FPGA. En [Marin-2007], [Marin-2009] se emplea una FPGA para implementar

un sistema de control remoto basado en manipuladores robóticos en red. La tarea de procesamiento de imagen se divide en dos flujos de datos principales. Por un lado, la imagen obtenida por la cámara situada en el extremo del robot se binariza (lo que representa una tarea de visión de bajo nivel) para, posteriormente, obtener descriptores de los objetos encontrados en la imagen (tarea de visión de medio nivel). Entonces, la interfaz de red de la FPGA se utiliza para enviar los momentos de los objetos utilizando el protocolo SNRP. Por otro lado, la imagen (transformada en una imagen de escala de grises) se combina con información visual del flujo de datos previo. Así, la imagen de salida de la FPGA es una imagen con realidad aumentada en la que se marca la posición del centroide de los objetos encontrados. Con el objetivo de realizar el seguimiento de un objeto en la imagen usando una cámara pan-tilt, Pérez describe una implementación sobre FPGA del cálculo del centro de gravedad de un objeto en la imagen en tiempo de ejecución [PérezC-2008]. Estas características visuales alimentan un esquema de control visual. En [Ata-2014] se propone una nueva aproximación para resolver problemas bien conocidos en la automatización industrial, como el control de calidad o la paletización. El sistema de paletización utiliza un sistema de visión embebido en una FPGA.

La segmentación de una imagen es un proceso integral para cualquier proceso de análisis de imagen. Es posible encontrar un gran número de aplicaciones para el análisis de imágenes médicas, especialmente en la localización de tumores. En [Sudha-2015] se presenta un método de segmentación para la extracción de imágenes intestinales basado en la segmentación por umbralización seguida de una técnica morfológica. La segmentación de las imágenes intestinales se obtiene usando el software de tiempo real Aphelion Dev. La imagen resultante se implementa en una FPGA Virtex de Xilinx que proporciona bajo consumo (el consumo total de la propuesta es de 0.182 W) y mínimos recursos hardware, tal como exigen las tareas de análisis en aplicaciones médicas de tiempo real.

Los telescopios también tienen que tratar con muchos problemas relacionados con el procesamiento de imágenes de tiempo real. Los grandes telescopios modernos requieren ópticas adaptativas. La turbulencia atmosférica se debe compensar en tiempo de ejecución, lo que requiere gran cantidad de potencia de procesamiento. Para resolverlo, en

[Rodríguez-Ramos-2006] se resumen los primeros resultados de un telescopio real con óptica adaptativa basado en FPGA. El sistema se ha instalado en el telescopio OGS del Observatorio del Teide en Tenerife. Este sistema está embebido en una Virtex-4 de Xilinx. En [Rodríguez-Ramos-2008], [Martin-2010] se presenta el diseño conceptual de un procesador de pendiente basado en FPGA para los sensores de frente de onda de estrellas guía de láser de telescopios extremadamente grandes. Los principales conceptos involucrados son el uso de subapertura como el grano más fino para el procesamiento paralelo, la necesidad de un procesador de flujo diferente para cada salida del detector y el uso de la fila de subaperturas (o subconjunto equivalente) para determinar la reutilización del procesamiento hardware. En [Rodríguez-2008], los mismos autores desarrollan un recuperador de fase FPGA para su cámara CAFADIS. El recuperador de fase diseñado lleva a cabo los cálculos dentro del tiempo característico atmosférico utilizando un muestreo realmente alto. Una transformada de Fourier rápida bidimensional se implementa sobre la arquitectura FPGA como algoritmo central del recuperador.

En sistemas de control visual, a menudo, se utiliza un patrón compuesto de formas geométricas para determinar la posición y orientación de los objetos. La Figura 4.6 muestra diferentes patrones utilizados habitualmente en este tipo de tareas.

En esta Tesis, se asumirá que los patrones consisten en un conjunto de círculos y que el controlador del sistema de control visual necesita cierta información de estos círculos, como la posición, el tamaño o la orientación. Esta es una tarea de visión de nivel medio. Para conseguirlo, se utilizarán dos algoritmos distintos, un algoritmo de detección de objetos visuales basado en el algoritmo de etiquetado de componentes conexas de pasada doble y una evolución que permitirá obtener mejores tiempos de proceso en el que se

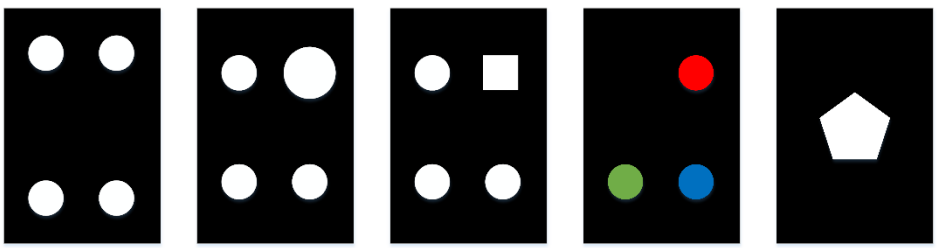


Figura 4.6 Patrones típicos para tareas de control visual

consigue etiquetar las componentes conexas en una sola pasada [Bailey-2011]. Este algoritmo recibe como entrada la imagen binarizada que contiene ya segmentado el patrón de círculos. La salida del algoritmo es el conjunto de centros de gravedad de los círculos en la imagen. En la siguiente fase del controlador, estos centros de gravedad se utilizarán como características visuales del sistema de control visual para calcular los pares que se deben enviar a los motores del robot.

4.1.3 Tareas de visión de alto nivel

La visión de alto nivel interpreta la escena a través de tareas específicas tales como razonamiento relacional, construcción de conocimiento, reconocimiento de objetos, etc. Una tarea en este grupo es una tarea de decisión basada en visión. La característica más importante de un dispositivo FPGA para este tipo de operaciones es el bajo consumo de energía. Las tareas de alto nivel son tareas de decisión que pueden reducir los requisitos de transmisión de datos de los sistemas de visión artificial. La adición de algoritmos de alto nivel a un sistema de visión artificial es una gran mejora para sensores remotos, como los embebidos en un satélite.

La imagen hiperespectral es una técnica que intenta identificar características en la superficie de la Tierra utilizando sensores que generalmente proporcionan grandes cantidades de datos. Normalmente, estos datos suelen ser recogidos por un satélite o un instrumento aerotransportado y enviados a una estación terrestre que lo procesa. De este modo, el ancho de banda de la conexión entre el satélite y la estación limita la información que se puede enviar y procesar en tiempo real. Un sistema de visión artificial a bordo que calcule las grandes cantidades de imágenes en tiempo real aumenta considerablemente el rendimiento del sistema [González-2012b]. De esta forma, el satélite sólo debe enviar la información importante, y no todas las imágenes para que se procesen en la estación terrestre. El trabajo presentado en [González-2012b] integra el algoritmo Winter N-FINDR [Winter-1999] en una FPGA para identificar los píxeles que definen distintas superficies. En [González-2009], [González-2010], [González-2013]) González et al. implementan el algoritmo PPI (del inglés *Pixel Purity Index*) sobre una FPGA para obtener estos puntos de interés en el suelo fotografiado por el satélite. Más adelante, en [González-2013],

desarrollan un diseño paralelo basado en FPGA del algoritmo de reconstrucción espacial (ISRA) para resolver el mismo problema de detección de superficies usando sensores de imágenes hiperespectrales.

En [Guzmán-2004] se describe otra tarea de visión de alto nivel relacionada con la fotografía de satélite. La principal contribución de este trabajo es el diseño de una aproximación FPGA capaz de predecir el error de reconstrucción de una imagen cuando se comprime con diferentes técnicas a una relación de compresión fija. Es decir, puede asesorar al sistema de compresión a bordo sobre qué tipo de algoritmo de compresión es más adecuado para los requisitos del satélite. En la mayoría de los casos, este coprocesador decidirá si el sistema de compresión JPEG2000 incorporado debe aplicar el algoritmo lossless o lossy. A veces, cuando se requiere una tarea de procesamiento de visión de alto nivel, el diseño de hardware implementa un microprocesador embebido en la FPGA (por ejemplo, Microblaze de Xilinx) que podría ejecutar un algoritmo programado en C y ejecutarse sin restricciones apreciables desde una aplicación de consola en un PC de escritorio. En [Vicente-2009] se emplea esta técnica para un sistema de visión artificial embebido para rastrear y contar personas. A veces, la FPGA se utiliza en un sistema de visión sólo para controlar el flujo de datos de imagen sobre procesadores DSP específicos. En [SánchezJ-2012] se desarrolló la arquitectura hardware de un sensor de vídeo inteligente utilizando dos procesadores DSP y un FPGA que controla, de manera flexible, la interconexión entre procesadores y el flujo de datos de imagen. El sensor de vídeo procesa las imágenes localmente para extraer los objetos de interés y clasificarlos.

La interacción hombre-máquina se está convirtiendo en una parte cada vez más importante de nuestra vida cotidiana. Las técnicas de detección de la mano, probablemente el método de interacción hombre-máquina más natural, han atraído cada vez más la atención en el campo de los sistemas de visión artificial y del reconocimiento de patrones, debido a su inestimable potencial en aplicaciones de interacción hombre-máquina. En [WangZ-2015] se propone una implementación sobre una FPGA Cyclone II de Altera de un algoritmo para el seguimiento de la mano y el reconocimiento de gestos. Para el seguimiento de la mano se utiliza un algoritmo de segmentación de la mano combinando el espacio de color YCbCr y un algoritmo de crecimiento de regiones para

una separación válida de la región, y luego se aplican a este resultado operaciones morfológicas incluyendo erosión y dilatación (que representa una tarea de visión de bajo nivel) para eliminar el ruido que pudiera quedar en la imagen procesada. Para detectar la mano en movimiento se utiliza el método de detección de movimiento por diferencia de tres imágenes (una tarea de visión de nivel medio). Finalmente, se puede localizar el centro de masas de la mano. Los autores desarrollaron un modelo de mano basado en el estado del dedo, que ayudó a confirmar el gesto de la mano al juzgar si los dedos están abiertos o cerrados.

Algunos sistemas de control visual emplean patrones compuestos por una combinación de diferentes formas geométricas (círculos, triángulos, cuadrados, etc.) para determinar la orientación de un objeto. Este tipo de patrones requiere una técnica de reconocimiento de formas además del algoritmo de detección de objetos en la imagen. En esta Tesis se usa un patrón de una única forma, ya que el controlador sólo necesita información sobre las coordenadas de su centro de gravedad, por eso no son necesarias tareas de decisión o alto nivel.

4.1.4 Sistemas de visión embebidos en FPGA para la detección de objetos

La división de los distintos algoritmos de visión en tareas de bajo, medio y alto nivel permite diferenciar los requisitos para su implementación, así como la complejidad de los algoritmos en sí mismos. Sin embargo, el sistema de visión artificial que se necesita implementar para un controlador visual como el que se propone en esta Tesis requiere el uso de tareas de visión de bajo y medio nivel. El sistema a implementar requerirá por tanto una mezcla de algoritmos de niveles distintos. El objetivo es obtener los centros de masas de los objetos visuales o componentes conexas encontrados en cada momento en la imagen. Para ello, puede ser necesario realizar operaciones previas de bajo nivel para ajustar, por ejemplo, el contraste y el brillo o realizar un filtrado o simplemente una erosión para eliminar ruido. Pero también es necesario realizar tareas de nivel medio que permitan etiquetar las componentes conexas u obtener sus centros de masas. La integración en una FPGA de muchas de estas operaciones ha sido ampliamente estudiada en la literatura. En [Bailey-2011] puede encontrarse gran cantidad de algoritmos de visión

de bajo nivel e incluso alguno de nivel medio embebidos todos en FPGA. La optimización de estos algoritmos gracias a la FPGA se debe sobre todo a la visión de la imagen como un vector que se recibe secuencialmente a través del digitalizador de vídeo. El trabajo sobre este flujo de datos visuales permite entonces obtener nuevos valores de intensidad de los píxeles tal y como llegan a la FPGA. Sobre todo, en operaciones de punto o en operaciones donde las memorias intermedias permiten evaluar un entorno determinado de un píxel (como podría ser una convolución, una erosión o un filtrado).

La visión embebida es un subconjunto de la visión artificial centrado en sistemas en los que los sensores son a menudo encapsulados en el dispositivo que controlan. El procesamiento se realiza a bordo, y el diseño del sistema requiere un equilibrio constante entre los algoritmos de uso intensivo de computación y el consumo de recursos. Las aplicaciones de visión embebida son diversas, y la implementación abarca desde sistemas de interacción para entretenimiento, como videojuegos [Camplani-2013] y sistemas de realidad virtual [KumarV-2015], hasta herramientas de asistencia humana, como robots y otros dispositivos autónomos [Hedge-2013]. En la industria del automóvil, los sistemas basados en sensores embebidos se centran comúnmente en tareas avanzadas como la detección de peatones [Besbes-2015], la prevención de salida del carril [Tapia-Espinoza-2013], [Eeum-2013] o la detección de obstáculos [ZhangY-2014]. En el campo de la vigilancia inteligente, la visión por ordenador se utiliza como un acelerador de análisis de vídeo. Los ejemplos incluyen la vigilancia del tráfico de vehículos [ChenYL-2011], vigilancia de eventos, así como sofisticados análisis biométricos (como, por ejemplo, cara o huellas dactilares) [Saeed-2015].

Los algoritmos para visión embebida más comunes tratan sobre la mejora de imagen, su segmentación, el reconocimiento de patrones o la detección de objetos. La detección de objetos se ocupa de la búsqueda y medición de parámetros de los objetos presentes en la escena. Esta tarea puede subdividirse en dos operaciones: segmentación del fondo y análisis de los componentes conexos. La segmentación del fondo implica la separación de información de fondo y de primer plano, mientras que el análisis de los componentes conexos se centra en la identificación de regiones de primer plano específicas que representan formas de objeto. El primer paso del análisis de objetos visuales es la

detección de estos objetos. La detección de objetos se utiliza para detectar conectividad entre objetos de la imagen (es decir, áreas de píxeles adyacentes que comparten un valor de color común). En las operaciones típicas de visión embebida, los principales cuellos de botella surgen debido a las altas transferencias de datos necesarias (información de la imagen almacenada en dispositivos de memoria), la implementación del algoritmo (serie o paralelo) y los recursos de hardware. Una posible solución para construir un sistema de visión embebida eficiente es desarrollar hardware personalizado dedicado a procesar la información de imágenes, para lo que resulta ideal el uso de dispositivos reconfigurables hardware como las FPGAs.

Sin embargo, sólo un pequeño subconjunto de algoritmos basados en software resulta adecuado para la implementación en hardware embebido, y pocas soluciones se han desarrollado para dispositivos de recursos limitados [Kyrkou-2011], [Viola-2004], [Mahlknecht-2004], [Patro-2014]. Esta Tesis se centra en el diseño de un algoritmo de detección de componentes conexas de una sola pasada, con alta optimización de sus recursos, tanto de puertas lógicas como de memoria. El algoritmo propuesto analiza la imagen de entrada a medida que se almacena desde la etapa de adquisición de imágenes. Sólo se almacena información relevante para la identificación de cada objeto encontrado en la escena.

Para la detección de las componentes conexas, una de las técnicas más empleadas es el algoritmo de Múltiples Pasadas. En [Crookes-1999] se implementa una aproximación del algoritmo de múltiples pasadas optimizando los recursos de una FPGA. El problema de este algoritmo es que se requiere más de una exploración de la imagen para completar el etiquetado. Para reducir el número de pases necesarios, y así mejorar el rendimiento del algoritmo, en [Jablonski-.2004] se propone la implementación en FPGA del clásico algoritmo de dos pasadas para el etiquetado de componentes conexas. Sin embargo, el método requiere una segunda pasada, y dos ciclos de reloj por píxel, más una pequeña sobrecarga debido a la fusión necesaria de regiones. Kofi et al. presentan un algoritmo de etiquetado de componentes conexos basado en el algoritmo de compresión de *Run-length* para su implementación sobre una FPGA [Kofi-2008], [Kofi-2010]. Se extraen las secuencias de las imágenes y se etiquetan inicialmente en la primera pasada, entonces se

resuelven los conjuntos equivalentes y las secuencias se traducen a componentes conexas. Sin embargo, la resolución de equivalencia sigue siendo secuencial y requiere que todas las secuencias se etiqueten al principio. Evidentemente, esto no lo hace adecuado para lograr el procesamiento en tiempo real de los datos transmitidos de la imagen. En [MaN-2008], [Johnston-2008] se describe un algoritmo de una sola pasada que requiere que se almacene únicamente una fila de etiquetas para determinar la etiqueta del píxel actual. Esto se logra extrayendo el vector de características para cada objeto para que no se requiera la imagen etiquetada. Se puede conseguir un rendimiento de hasta 1 píxel de imagen por ciclo de reloj. Esta arquitectura fue optimizada por Ma et al. aplicando un esquema de re-etiquetado agresivo, reutilizando recursos de memoria después de que se detecta el último píxel de un objeto en la secuencia de vídeo [MaN-2008]. Esto reduce el número de etiquetas a costa de recursos de hardware adicionales para traducir las etiquetas entre las filas. Estos trabajos de Man et al. han servido de base para nuevas mejoras en la optimización de recursos en este algoritmo de pasada simple [Klayber-2016]. Las optimizaciones que se realizan en estos trabajos buscan reducir la cantidad de memoria necesaria para que se puedan aplicar estos algoritmos a la búsqueda de objetos en movimiento para imágenes de alta resolución (como las cámaras 4K u 8K). Para los sistemas de control visual este no es el principal requisito. El principal requisito es la obtención de los centroides de los objetos encontrados en la imagen con la mayor rapidez posible y a ser posible en un tiempo predefinido, de forma que pueda establecerse el periodo del bucle de control. En [ZhaoF-2013] se implementa el algoritmo de una pasada, pero se utiliza un DSP para ciertos cálculos, mientras que otros cálculos se dejan para la FPGA. También es muy habitual el empleo de los procesadores software embebidos en FPGA. El uso de estos procesadores mejora el comportamiento de la implementación sobre un GPP, porque permiten que cierto hardware de la FPGA se utilice para otras tareas independientemente de las puertas lógicas y recursos utilizados por el procesador software. Sin embargo, esta capa intermedia de abstracción que proporciona el procesador software impide obtener la máxima optimización de los recursos paralelos de la FPGA. En [Acevedo-Avila-2016] se utiliza también un co-procesador embebido en la FPGA para etiquetar objetos y calcular su centroide y su área.

Otros trabajos como el presentado en [Bochem-2011] realiza la detección de los objetos en la escena para calcular luego sus centroides. La imagen pasa por diversas etapas, pero todas las etapas trabajan con su propio reloj y almacenan sus datos de resultado. Con lo que se desaprovecha el paralelismo de las FPGA.

El sistema que se propone en esta Tesis resuelve muchos de los inconvenientes de los métodos encontrados en la literatura, sobre todo de cara a obtener una respuesta en un tiempo predeterminado y lo más rápidamente posible en función del refresco de la cámara empleada por el controlador. Por ello, se propone el uso de una cámara rápida, que a una resolución de 752 x 582 es capaz de trabajar a 340 fps. Buscando aprovechar al máximo esta velocidad, se diseña un cauce segmentado que permite trabajar con el flujo de la imagen proporcionado por el digitalizador de vídeo. Es decir, no se requiere que la imagen se almacene en una memoria para luego empezar a procesarla. La imagen se va enviando a la FPGA como un vector, con la información del nivel de intensidad de cada píxel, tal y como se va capturando en el CCD. Cuando se llena el cauce, después de unas primeras iteraciones, se obtiene ya el resultado de operaciones como el redimensionado o la umbralización tal como llega el píxel. Si la operación es una erosión, se retrasa el tiempo requerido para almacenar dos filas, pero a partir de ese momento, de nuevo se tiene la información del píxel a la velocidad a la que se recibe. Lo mismo ocurre con el algoritmo de una pasada de detección de componentes conexas. Además, al mismo tiempo se almacenan valores de posición para en cuanto acabe esa fase de detección calcular el centroide mucho más rápidamente. Es importante recalcar que se trabaja con el vector de la imagen, no con la imagen almacenada en memoria entre fases. Las fases se realizan en paralelo. Tampoco se utilizan procesadores software implementados sobre la FPGA, que proporcionarían una gran ayuda para la implementación del algoritmo, pero que impedirían trabajar a bajo nivel y optimizar al máximo los recursos proporcionados por la FPGA.

4.2 Implementación del módulo de visión en la FPGA

Los dispositivos FPGAs son ideales para el procesamiento de imágenes, particularmente para tareas de bajo y medio nivel, donde se aprovecha especialmente el paralelismo que permiten estos dispositivos. Para una arquitectura basada en cauce segmentado, se construye un bloque de hardware diferente para cada operación de procesamiento de imagen. El bloque que implementa la operación de procesamiento de imagen pasa sus datos procesados al siguiente bloque, que realiza una operación diferente. Cuando el sistema no es síncrono, se requieren memorias intermedias entre operaciones. Estas memorias intermedias gestionan las variaciones en el flujo de datos. Como se ha comentado anteriormente, puede explotarse el paralelismo espacial con la construcción de copias múltiples de las operaciones implementadas y la asignación de diferentes particiones de la imagen a cada copia. Se puede lograr un paralelismo espacial completo construyendo un procesador para cada píxel. En la práctica, la alta resolución de imagen de las cámaras modernas hace que esto sea muy costoso en recursos de la FPGA.

El paralelismo lógico es el paralelismo general contenido en un programa, es decir, todos los cálculos que pueden, de acuerdo con la semántica del lenguaje de programación, ejecutarse en paralelo. El paralelismo lógico dentro de una operación de procesamiento de imágenes encaja en una implementación en una FPGA. Aquí es donde la mayoría de los algoritmos de procesamiento de imágenes pueden mejorar significativamente el rendimiento. Para ello, los bucles internos se despliegan. Por lo tanto, las operaciones se realizan en hardware paralelo en lugar de secuencialmente.

En esta Tesis, los objetos utilizados como características visuales para guiar al robot son objetos blancos con un fondo negro. Así, la imagen adquirida por la cámara tendrá un conjunto de objetos blancos, variando de uno a cuatro. Y el objetivo del módulo de visión es determinar las coordenadas de los centros de gravedad de los objetos encontrados en la imagen.

La Figura 4.7 muestra un esquema de la tarea de visión embebida en una FPGA implementada en esta Tesis. Incluye operaciones de procesamiento de imágenes de bajo y

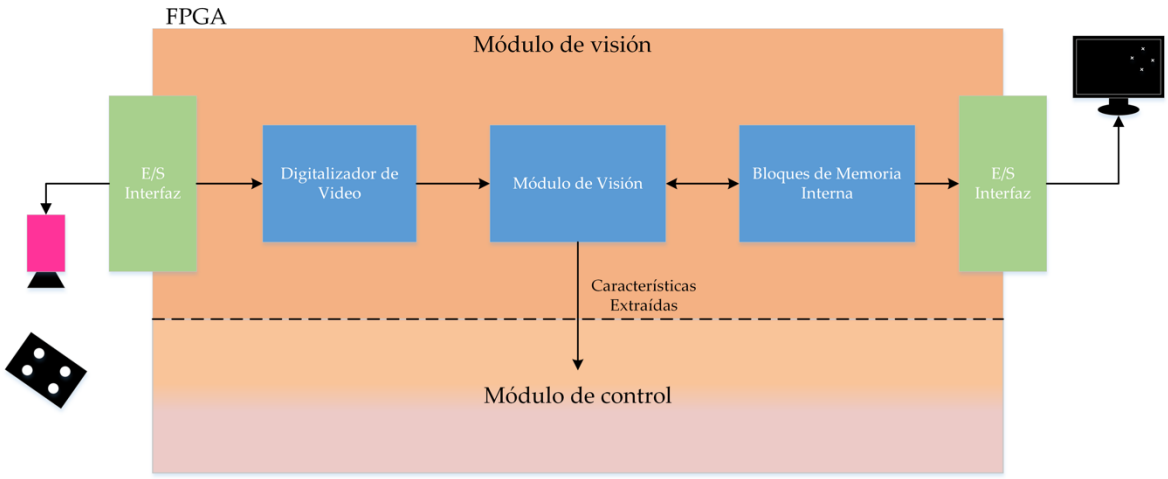


Figura 4.7 Esquema del sistema de visión artificial embebido en la FPGA. Seguimiento de un patrón y cálculo de su centro de gravedad a la frecuencia de captura de la cámara

medio nivel. Las posibilidades de paralelismo proporcionadas por la FPGA tienen efectos en la construcción del sistema de visión. La implementación de una arquitectura en cauce segmentado en un FPGA permite operar a la misma frecuencia que se sirven los píxeles. Dado que el consumo de energía está directamente relacionado con la frecuencia de reloj, una menor frecuencia implica una menor demanda de energía por parte del sistema. La tarea de visión descrita en la Figura 4.7 es una aproximación FPGA típica a una tarea de procesamiento de imágenes.

Normalmente, los datos de la imagen circulan en serie o de manera secuencial, lo que encaja perfectamente en una implementación hardware, especialmente si es posible interactuar directamente con la cámara. El sistema de visión está diseñado para tener, tanto como sea posible, una arquitectura abierta. Para conseguirlo, se divide en dos bloques funcionales, Digitalizador de vídeo (dependiente del hardware) y el módulo de Visión (que no depende del hardware) (ver Figura 4.7).

4.2.1 Digitalizador de vídeo

El bloque Digitalizador de vídeo realiza la comunicación con la cámara para recibir los píxeles que fluyen desde el sensor. Este bloque es responsable de implementar el protocolo requerido (*Camera Link*, *Gigabit Ethernet*, *I2C*, etc.) para comunicarse con el dispositivo de captura, configurarlo y obtener el flujo de imágenes. En esta Tesis, se utiliza una cámara de tipo *CameraLink*, y se implementa la interfaz de protocolo correspondiente.

Pero en el caso de utilizar otro tipo de cámaras sería necesario cambiarlo, es por esto que este es un bloque dependiente del hardware. Una vez configurado, e iniciada la transmisión de datos, el flujo de píxeles se introduce en el Módulo de Visión para ser procesado.

Para probar el sistema propuesto en esta Tesis, se hizo uso de una cámara MV-D752-160 CMOS de alta velocidad de Photonfocus. Como se ha comentado anteriormente, esta cámara dispone de una interfaz digital CameraLink. La cámara envía los datos de imagen a una frecuencia de 80 MHz, enviando dos píxeles al mismo tiempo cada ciclo de reloj. La Figura 4.8 muestra el diagrama de sincronización de las señales de la cámara en modo de ejecución libre.

4.2.2 Módulo de Visión

El módulo de Visión es un módulo de arquitectura completamente abierta, es decir, no depende del hardware y está compuesto por un conjunto de cinco etapas (ver Figura 4.9). Recibe los píxeles que fluyen desde el Digitalizador de vídeo, para extraer la información visual deseada. Los algoritmos de visión se implementan en un cauce segmentado. Así, cada píxel recibido por el módulo de visión se procesa cuando llega, sin la necesidad de

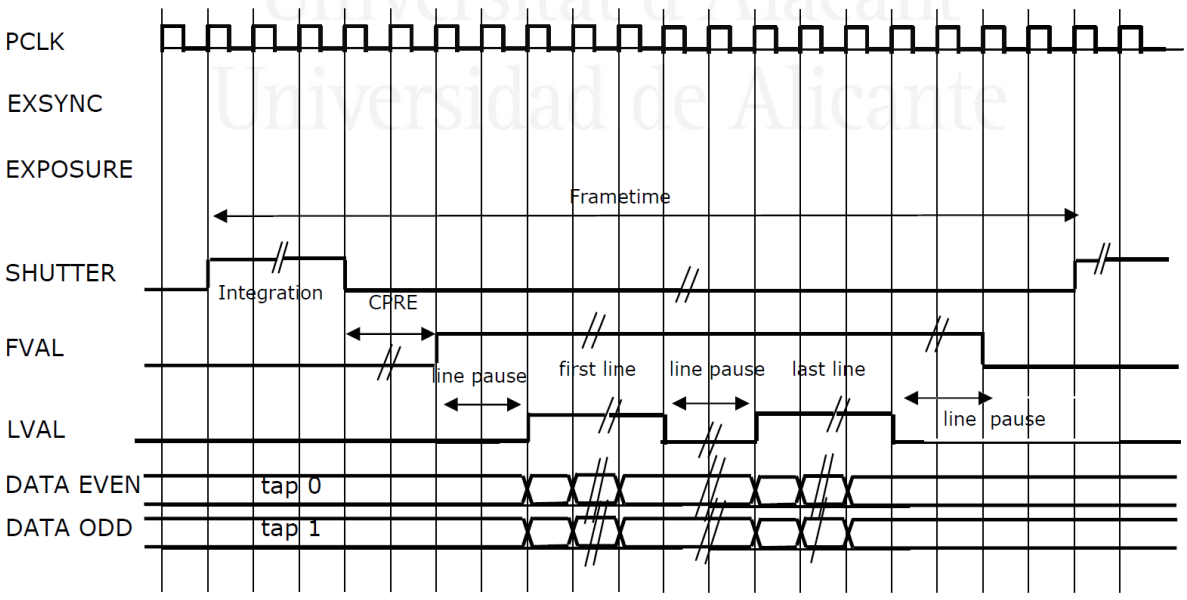


Figura 4.8 Diagrama de tiempo de la cámara Photonfocus MV-D752-160 en modo de ejecución libre (Fuente: [Photonfocus-2004])

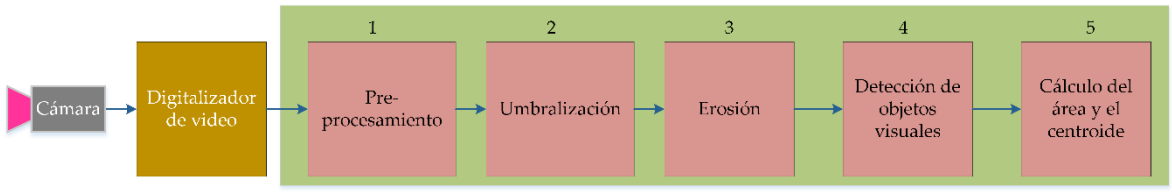


Figura 4.9 Esquema del módulo de Visión

almacenar toda la imagen en memoria intermedia antes de procesarla. Por lo tanto, la latencia del sistema completo se reduce, al tiempo que se ahorra una cantidad considerable de recursos de almacenamiento y tráfico de comunicación. La implementación de un cauce segmentado es especialmente útil para obtener una respuesta de tiempo reducido en un tiempo conocido. Esta es la característica más importante de la arquitectura propuesta con respecto a la tradicional (los GPP). Con la arquitectura propuesta en esta Tesis, el sistema de control visual se convierte en un sistema de tiempo real.

La Figura 4.9 muestra la funcionalidad de cada etapa del módulo de visión. El primer paso preprocesa la imagen para obtener un formato único de la imagen, independientemente de la cámara utilizada (por ejemplo, conversión de la imagen de RGB a escala de grises, redimensionamiento de la imagen, filtrado de ruido, etc.). En la segunda etapa, la imagen se binariza a través de un proceso de umbralizado. El resultado es una imagen binaria con todos los datos esenciales del patrón. El tercer paso es una operación de erosión. Se trata de una operación morfológica, que elimina los píxeles situados en el borde de cualquier objeto de la imagen. Esta etapa se utiliza principalmente para reducir el ruido de la imagen y puede usarse o no, dependiendo de los filtros aplicados en la etapa de preprocesamiento. La cuarta fase es la etapa más importante del módulo de visión: la detección de los objetos visuales. Esta es una tarea de visión de nivel medio. El objetivo de este módulo es detectar los píxeles conectados que forman un objeto en la imagen para posteriormente asignarles una etiqueta. En esta Tesis se proponen dos estrategias para la detección de estas componentes conexas. La primera estrategia se basa en el algoritmo de etiquetado de componentes conexas de doble pasada. La segunda es una evolución de la anterior que mejora claramente la latencia del sistema y optimiza los

recursos de la FPGA: el algoritmo de pasada simple. Finalmente, el módulo de visión tiene que proporcionar las coordenadas del centro de gravedad de los objetos visuales detectados en la etapa 4 (las características visuales) al siguiente módulo (el módulo de Control que se verá en el Capítulo 5). El quinto submódulo (cálculo de área y centroide) realiza esta tarea.

4.2.2.1 Preprocesamiento

En esta etapa se aplican operaciones preliminares para lograr un formato único de imagen independiente de la cámara utilizada, como la conversión de un espacio de color a otro (de RGB a escalas de gris por ejemplo), redimensionado de la imagen, filtrado de ruido, etc.

El objetivo de esta etapa es el de preparar adecuadamente la imagen para las siguientes fases de procesamiento. La cámara Photonfocus MV-D752-160 usada en esta Tesis, proporciona 2 píxeles en cada ciclo de reloj generado por la cámara, como demuestra la Figura 4.8, algo que no es habitual en las cámaras digitales y complica bastante la implementación de los algoritmos de procesamiento de imagen en la FPGA. Por lo tanto, se ha empleado la etapa de preprocesamiento para modificar el flujo de datos de la imagen recibido de la cámara para proporcionar un solo píxel en cada ciclo de reloj a los siguientes módulos de visión. La Figura 4.10 muestra el método empleado para conseguir este objetivo. La idea es dividir la imagen en bloques de 2x2 píxeles, y fusionar los cuatro píxeles de cada bloque en un píxel cuyo valor es el promedio de los 4 píxeles del bloque. Implementar este método en la FPGA requiere una memoria RAM configurada como FIFO (del inglés *First In First Out*) para almacenar los datos de una fila de la imagen y usar su contenido para calcular los promedios con la siguiente fila.

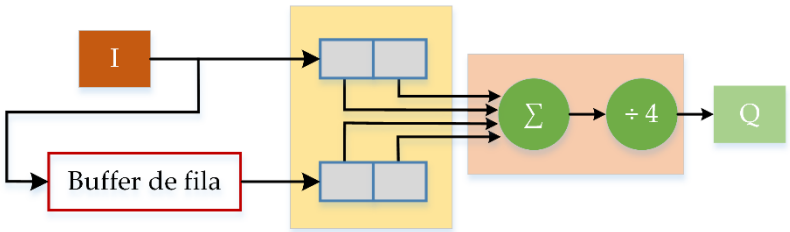


Figura 4.10 Algoritmo de preprocesamiento implementado en la FPGA

La Figura 4.11 muestra un ejemplo del método explicado arriba, donde la primera fila se almacena en el buffer de fila y sus datos se utilizan mientras se reciben los datos de la segunda fila, calculando el promedio que será el valor del píxel de salida. Utilizando este método podemos destacar dos consecuencias importantes, la primera es que la resolución de la imagen de salida es la mitad de las dimensiones de la imagen de entrada y, la segunda es, que este método actúa como un filtro de media, que sirve para reducir el ruido en la imagen.

4.2.2.2 Umbralización

La umbralización, en su forma básica, compara cada píxel de la imagen con un nivel de umbral y asigna la salida a verdadero o falso (blanco o negro), como se muestra en la Figura 4.12. Puesto que cada píxel se trata de forma individual, la umbralización es una operación puntual, que no requiere recursos de almacenamiento, ni realizar operaciones matemáticas, y no supone retraso temporal en el cauce segmentado.

$$Q(x,y)=\begin{cases}1, & I(x,y)\geq Umbral\\ 0, & I(x,y)<Umbral\end{cases} \tag{4.1}$$

La Figura 4.13 es un ejemplo de la umbralización durante 3 ciclos de reloj. La imagen resultante de esta operación es una imagen binaria en la que cada píxel se representa con uno o cero.

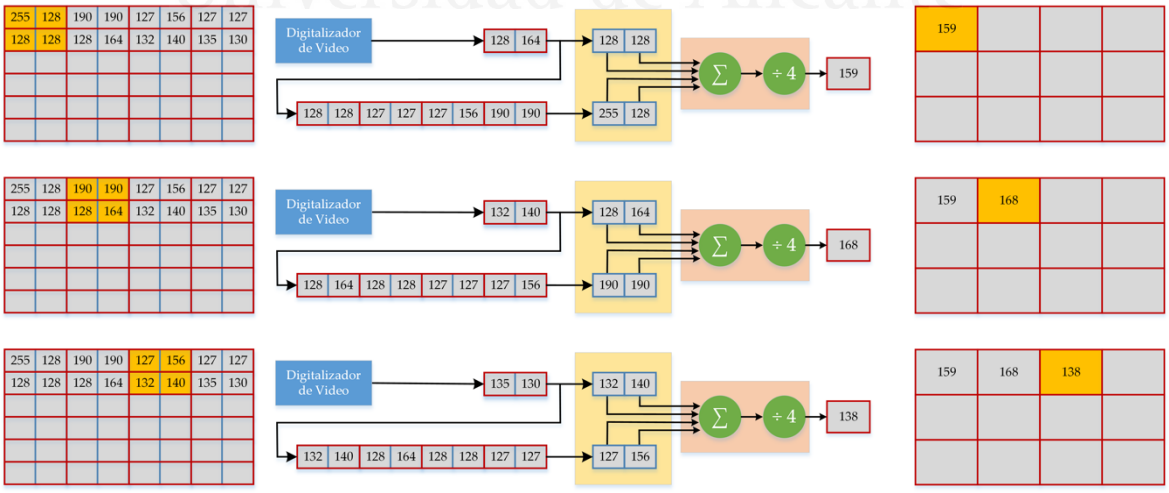


Figura 4.11 Ejemplo práctico de la ejecución del algoritmo de preprocesamiento propuesto

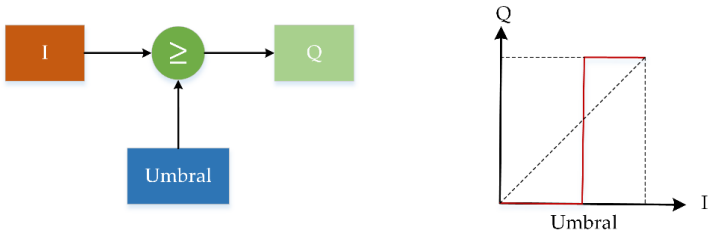


Figura 4.12 Operación de punto: la umbralización

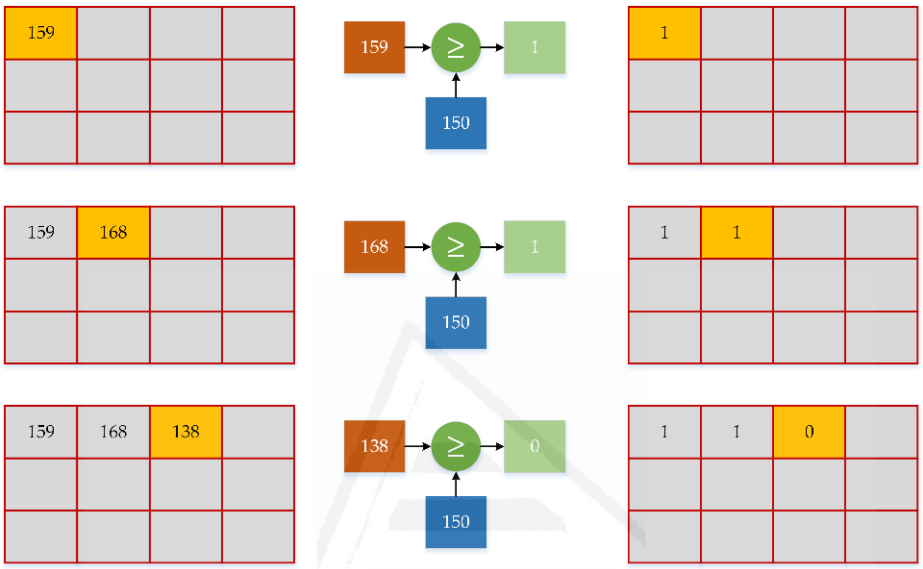


Figura 4.13 Ejemplo práctico de la ejecución de la umbralización

4.2.2.3 Erosión

La erosión es una de las operaciones básicas del filtrado morfológico que se aplica sobre una imagen binaria *I*, utilizando un elemento estructurante *S*, que define la forma de realizar el filtrado. Una imagen binaria considera que cada píxel pertenece a una de dos clases: objeto y fondo. En general, los píxeles del objeto están representados por un 1 y el fondo está representado por 0. El elemento estructurante, *S*, consiste en un conjunto de vectores o desplazamientos. Por lo tanto, se puede considerar como otra imagen binaria o forma. Con la erosión, un píxel del objeto se mantiene sólo si el elemento estructurante se encaja completamente dentro del objeto. Considerando el elemento estructurante como una ventana, la salida se considera un píxel del objeto solamente si todas las entradas son 1. Con otras palabras, la erosión es, una operación AND lógica de los píxeles dentro de la ventana (ver Figura 4.14).

$$Q(x,y) = I \Theta S = \wedge_{i,j \in S} I[x+i, y+j] \tag{4.2}$$

Se llama erosión, porque el tamaño del objeto se hace más pequeño como resultado de la aplicación de la operación morfológica. Como se puede observar en el ejemplo de la Figura 4.15, empleando un elemento estructurante de 3x3, los bordes del objeto grande se han borrado mientras los objetos pequeños han desaparecido completamente.

En la presente Tesis, se emplea la erosión con un elemento estructurante de 3x3 como filtro de ruido, ya que el patrón visual usado es un conjunto de entre 1 y 4 formas geométricas, como por ejemplo círculos o cuadrados. Así, eliminar el perímetro de una forma con 1 píxel de profundidad no afecta en absoluto en la posición del centro de dicha forma, pero se logra eliminar gran parte del ruido de la imagen de forma rápida y eficaz. La erosión, como un filtro local, amplía las operaciones de punto (como la umbralización) al tener la salida una función de los valores de píxeles dentro de un vecindario o el

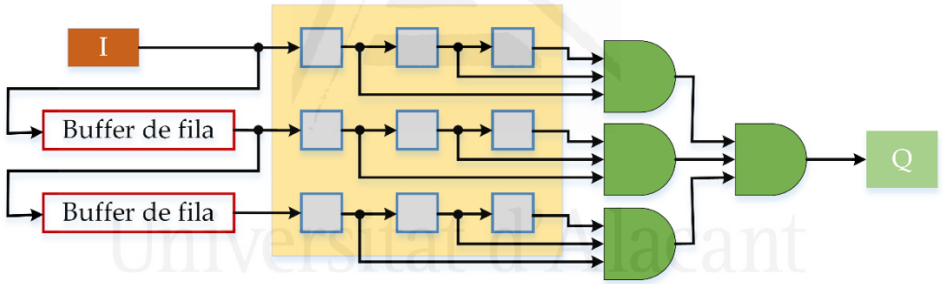


Figura 4.14 Operación morfológica: la erosión. Con una ventana de 3x3

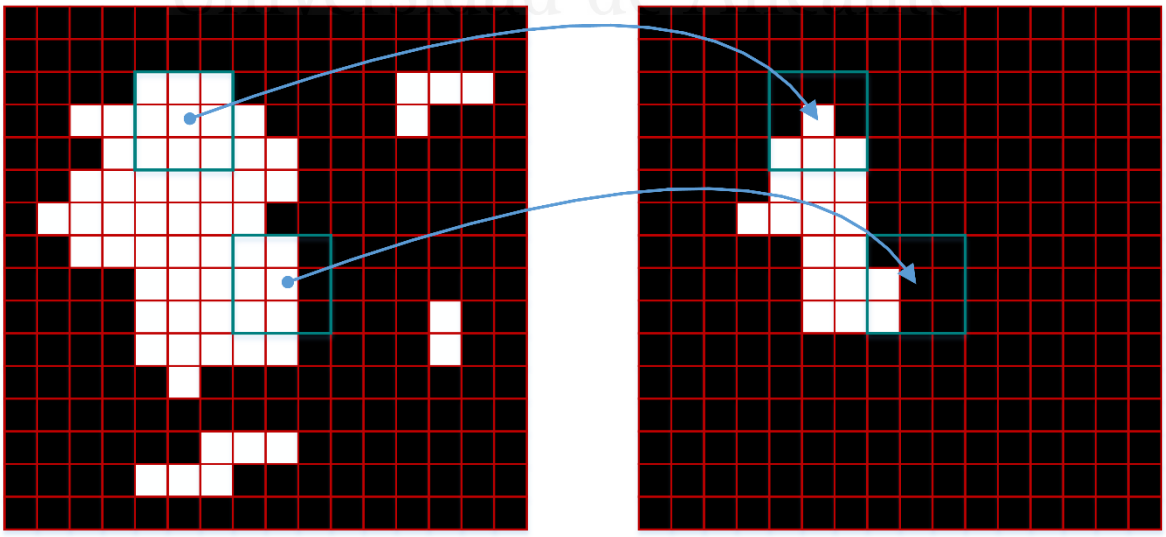


Figura 4.15 Ejemplo práctico de la ejecución de la erosión

elemento estructurante. Por tanto, hace falta algún tipo de *buffers* o memorias intermedias para guardar los valores de píxeles necesarios para operaciones de filtrado posteriores. En el caso de usar un elemento estructurante de 3x3, es necesario almacenar los datos de las últimas 2 filas de la imagen de entrada en buffers tipo FIFO (ver Figura 4.14), para usar sus valores mientras se recibe la siguiente fila. Esto causa que el procesamiento de imagen en el cauce segmentado sufra un retardo equivalente al tiempo que tarde el módulo en recibir dos filas de la imagen.

4.2.2.4 Detección de objetos visuales

En visión artificial un objeto visual se considera como un conjunto de píxeles adyacentes que describen un atributo común. Este atributo se define en la mayoría de los casos por el color o el brillo del píxel. La representación del píxel depende del modelo de color aplicado en el material de la imagen. Para la detección de los objetos visuales, el primer problema que se debe resolver es la identificación de los píxeles relevantes. En este trabajo se considera que un píxel es relevante si su valor de brillo excede un valor de umbral especificado. Este valor umbral puede modificarse a través de un encoder incremental situado en la placa de la FPGA, y se corresponde con la fase descrita en el Apartado 4.2.2.2 de Umbralización.

El módulo de detección de objetos visuales es la etapa más importante en el diseño del módulo de visión. Esto se debe a que su implementación requiere la mayor cantidad de recursos de hardware disponibles en la FPGA. Dos son los objetivos de este bloque: por un lado, la detección de los píxeles conectados de un objeto; y por otro lado asignarles una etiqueta. En la literatura hay muchos algoritmos para detectar objetos visuales en una imagen [Bailey-2011], por ejemplo, codificación *Run-Length*, codificación en cadena, pasada simple y pasada múltiple, etc. En el sistema de visión artificial que se propone en esta Tesis se implementan dos métodos para la detección de objetos visuales: el método de detección de componentes conexas de dos pasadas, y el método de detección de componentes conexas de pasada simple.

4.2.2.4.1 Método de detección de dos pasadas

Para optimizar la implementación en una FPGA, se eligió en un principio un algoritmo de etiquetado de dos pasos. Este algoritmo realiza el etiquetado de los objetos de la imagen a través de dos etapas. En la primera etapa, se procesan los píxeles del módulo anterior (Erosión o Umbralización) y se les asignan etiquetas temporales. Estos datos se almacenan en la RAM. Al mismo tiempo, se llena una LUT (tabla de búsqueda). Esta tabla (llamada tabla LUT equivalente) determina qué etiqueta pertenece a qué objeto. Esto se utiliza en la segunda etapa para interpretar los datos de la imagen provisionalmente etiquetados, y fusionar todas las etiquetas que pertenecen al mismo objeto. La Figura 4.16 muestra la funcionalidad de cada etapa del algoritmo mencionado. La idea básica de la primera etapa es: si el píxel recibido no es cero (el valor de los píxeles del fondo), se examinan los píxeles superiores "A", "B" y "C" (de la fila anterior) y el píxel anterior en la misma fila "D". Si todos estos píxeles son cero, se asigna una nueva etiqueta al píxel actual. Si este no es el caso, la etiqueta con el valor más pequeño (distinto de cero) de los píxeles vecinos "A", "B", "C" y "D" se asigna al píxel real. Después de procesar toda la imagen y etiquetarla en la memoria intermedia, se almacena una copia de la tabla LUT equivalente. De este modo, la primera etapa puede iniciar el procesamiento de una nueva imagen mientras la imagen anterior se sigue procesando en la segunda etapa (en el cauce segmentado). La tarea de la segunda etapa es bastante simple: leer los píxeles almacenados en la memoria RAM e interpretarlos de acuerdo a la tabla LUT equivalente. En la Figura 4.17 se muestra un ejemplo práctico del algoritmo descrito para la detección de objetos con el algoritmo de dos pasadas descrito.

Este algoritmo permite resolver de manera paralela el etiquetado de las componentes conexas de una imagen. Sin embargo, la segunda fase requiere almacenar la imagen en RAM, con lo que se corta el paralelismo en este módulo, ya que el resultado de la primera fase se debe tener para poder empezar con la segunda fase. Es decir, la fase 1 y la fase 2 son secuenciales.

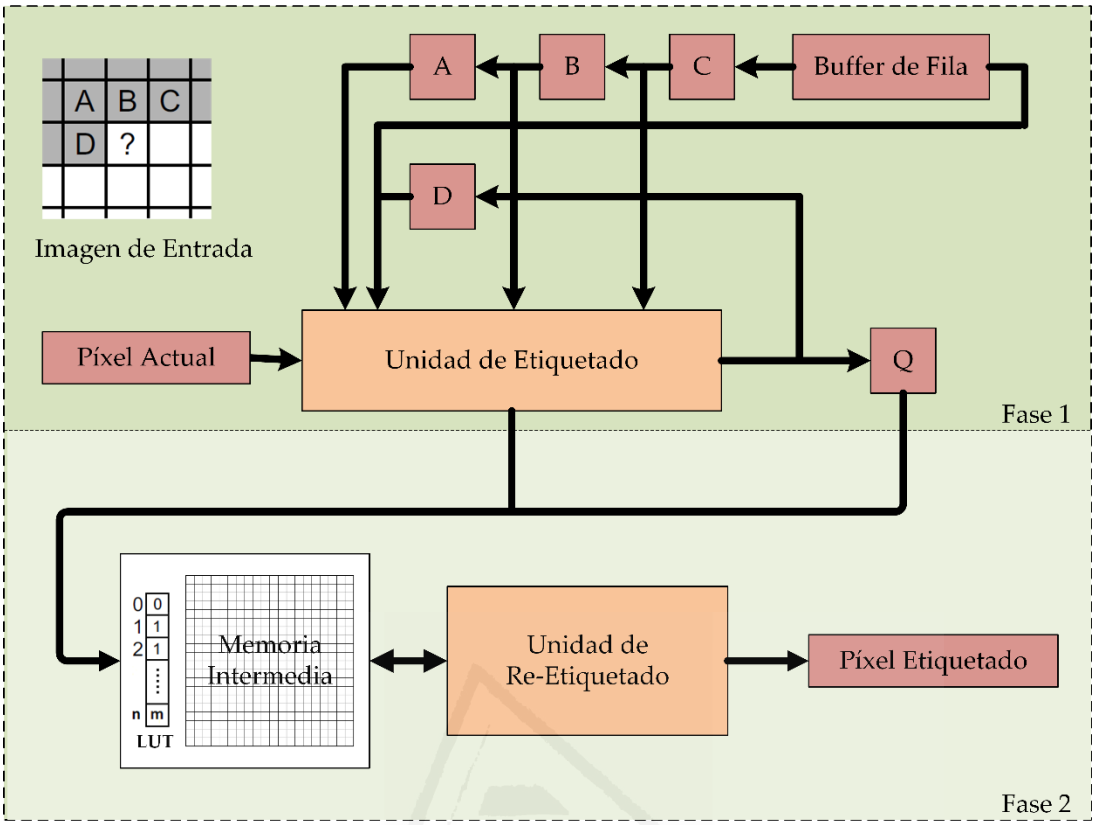


Figura 4.16 Algoritmo de Pasada Doble: detección de objetos

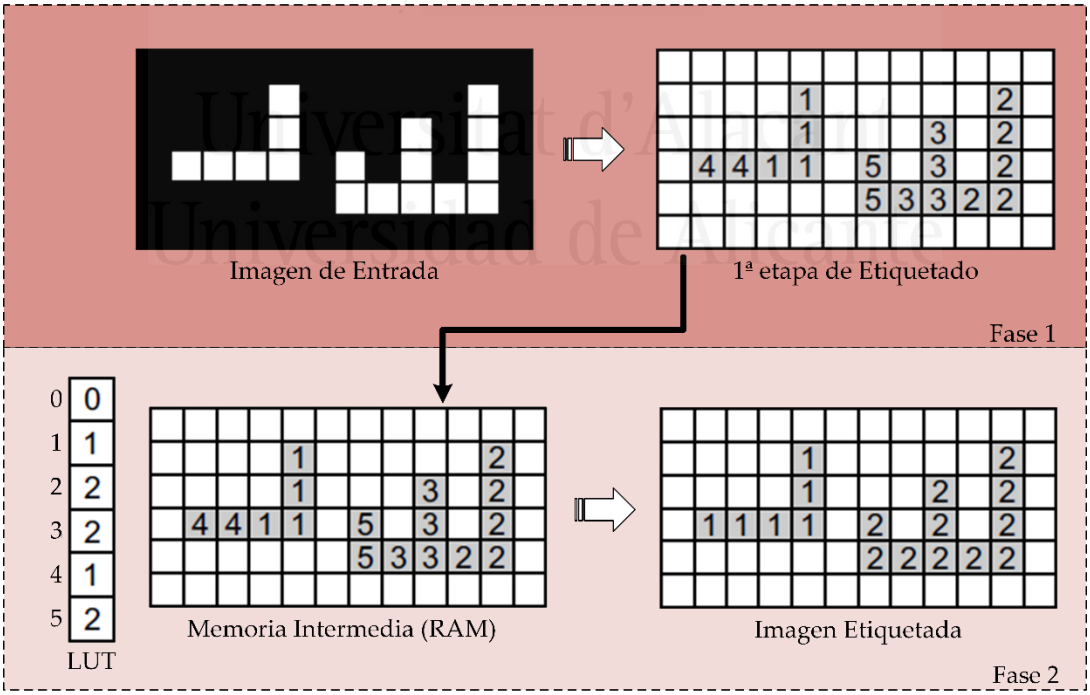


Figura 4.17 Algoritmo de Pasada Doble: ejemplo práctico del algoritmo de detección de objetos

4.2.2.4.2 Método de detección de pasada simple

Con el objetivo de mejorar la implementación del algoritmo anteriormente descrito para la detección de objetos visuales, se muestra a continuación una modificación del algoritmo de Pasada Simple, que permite realizar la detección en una sola pasada. Este algoritmo se ha adaptado a los requisitos del sistema propuesto y es capaz de realizar el etiquetado de los objetos de la imagen en una sola fase. La funcionalidad de este algoritmo es bastante simple. Se procesan los píxeles recibidos del módulo anterior (Erosión) y se les asignan etiquetas temporales. Al mismo tiempo, se va rellenando una tabla LUT (tabla de búsqueda). Esta tabla contiene un registro (fila) para cada etiqueta detectada en la imagen. Cada fila tiene la siguiente información sobre su etiqueta: si está conectada con otra etiqueta (etiqueta equivalente); número de todos los píxeles que están etiquetados con esta etiqueta en la imagen; y la suma de las coordenadas X, y de las coordenadas Y de estos píxeles. Esta información se utiliza para calcular el centroide de los objetos detectados al final de cada fotograma. Las dos últimas columnas se utilizan para acumular la posición X y la posición Y de todos los píxeles que están conectados entre sí y forman un objeto. Puesto que la definición del centro de gravedad (o centro de masas) de un objeto visual en una imagen binaria se define como la posición media X e Y del objeto binario, se define como un punto, cuyo valor X se calcula sumando las coordenadas X de todos los píxeles en el objeto, y luego dividiéndose por el número total de píxeles. Se realiza el mismo procedimiento para el valor Y. La Figura 4.18 muestra el funcionamiento de este algoritmo. La idea básica de esta etapa es: si el píxel de lectura no es cero (con lo que sería un píxel del fondo), se examinan los píxeles superiores "A", "B" y "C" (de la fila anterior) y el píxel anterior en la misma fila "D". Si todos estos píxeles son cero, se asigna una nueva etiqueta al píxel actual. Si no es así, se asigna la etiqueta con el valor más pequeño (distinto de cero) de los píxeles vecinos "A", "B", "C" y "D". Para cada nueva etiqueta detectada en la imagen, se agrega un nuevo registro a la tabla LUT. Mientras que si el píxel actual tiene etiquetas vecinas diferentes (es el caso del píxel rojo en la Figura 4.19), las dos etiquetas diferentes se combinan actualizando el valor más alto de etiqueta equivalente en la tabla LUT. La Figura 4.19 muestra un ejemplo práctico del algoritmo descrito para la detección de objetos.

Al final de cada trama de imagen, el paso final del nuevo módulo propuesto de Detección de objetos y cálculo de centroides es calcular el centro de gravedad de cada objeto visual encontrado, utilizando los datos finales actualizados en la tabla LUT. El centro de masas de los objetos detectados en la imagen es la salida del módulo de visión y al mismo tiempo es la entrada al módulo de control que se describirá en el Capítulo 5. El método utilizado para calcular el centroide del objeto tiene en cuenta la posición de todos sus píxeles. Logra una mejor resolución en el cálculo del centro de gravedad que otros enfoques como el método del *Bounding Box*, debido al hecho de que este último se ve mucho más afectado por el ruido de la imagen [Bochem-2011].

Cabe mencionar que el centro de masas se calcula en paralelo con la operación de etiquetado de los píxeles, contando y añadiendo las coordenadas X e Y del píxel actual a la fila LUT correspondiente a la etiqueta del píxel. Una vez que la trama de la imagen se procesa completamente, todas las etiquetas LUT que tienen la misma etiqueta equivalente (objetos conectados) se consideran un solo objeto.

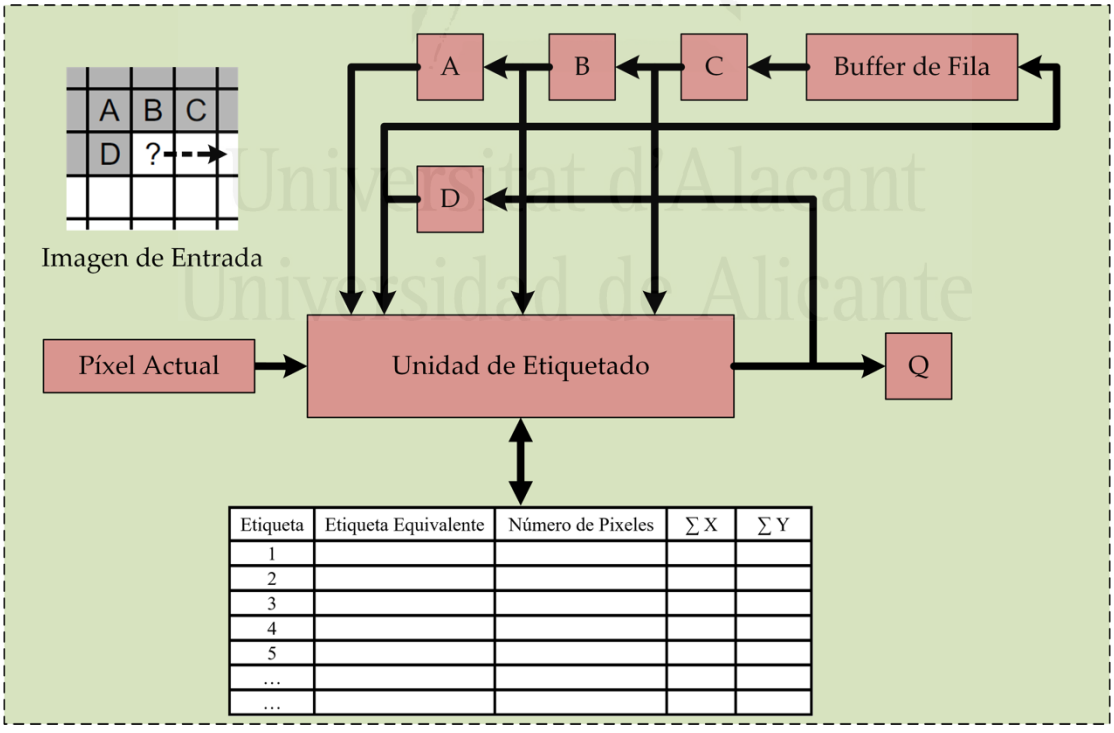


Figura 4.18 Algoritmo de Pasada Simple: detección de objetos y algoritmo de cálculo del centro de gravedad

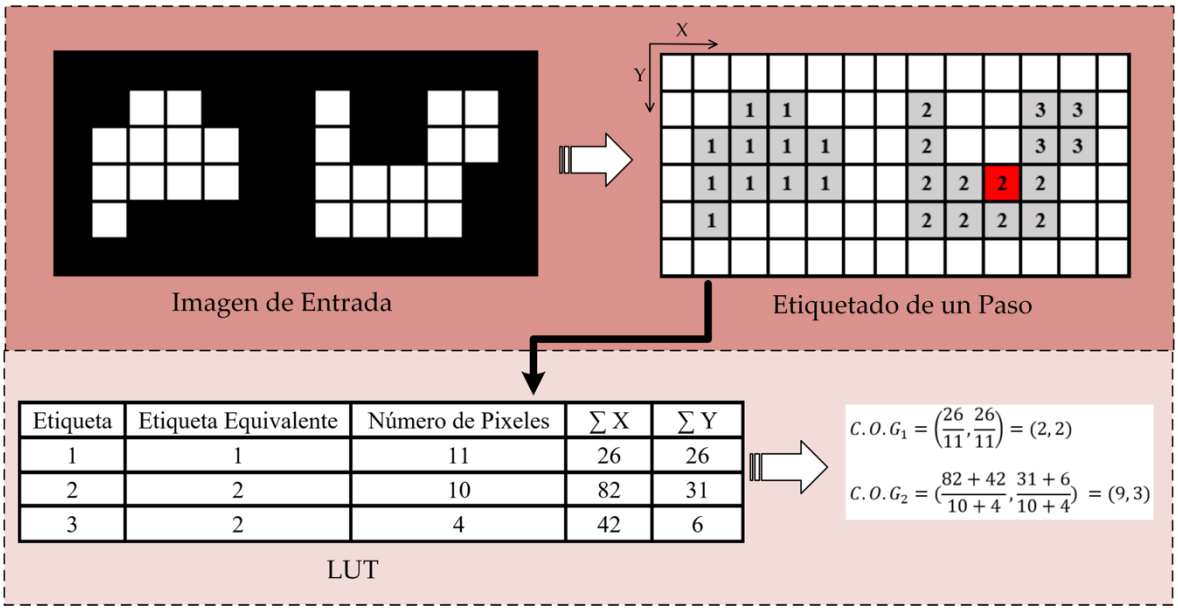


Figura 4.19 Algoritmo de Pasada Simple: ejemplo práctico del algoritmo de detección de objetos

4.2.2.5 Cálculo del centroide

Este módulo se utiliza únicamente para la implementación con el algoritmo de dos pasadas descrito en el Apartado 4.2.2.4.1. Una vez que los píxeles de la imagen están etiquetados, la etapa final del módulo de visión calcula el centro de gravedad de cada objeto. Como se ha mencionado anteriormente, estos datos son la salida del módulo de visión y también la entrada para el módulo de control que se describirá en el Capítulo 5. El módulo que se describe a continuación se ejecuta en paralelo con la fase 2 del método de pasada doble. Para alcanzar este objetivo se han implementado dos enfoques en la FPGA:

a) *Bounding Box*: El cálculo basado en el *bounding box* estima el punto central del objeto buscando las coordenadas XY mínimas y máximas del objeto. Este cálculo es muy eficiente y no causa grandes problemas de rendimiento.

$$x_{centro\_bb} = \frac{\max\_X_{posición} + \min\_X_{posición}}{2} \tag{4.3}$$

$$y_{centro\_bb} = \frac{\max\_Y_{posición} + \min\_Y_{posición}}{2} \tag{4.4}$$

b) *Centro de gravedad*: Con esta técnica, las coordenadas de todos los píxeles del objeto detectado se tienen en cuenta para el cálculo del punto central. El algoritmo combina los valores de las coordenadas de los píxeles detectados como una suma ponderada y calcula una media de las coordenadas del centro.

$$x_{centro\_CDG} = \frac{\sum \mathrm{X}_{\text{posición}} \text{Para todos los píxeles del blob}}{\text{Número de píxeles del blob}} \tag{4.5}$$

$$y_{centro\_CDG} = \frac{\sum \mathrm{Y}_{\text{posición}} \text{Para todos los píxeles del blob}}{\text{Número de píxeles del blob}} \tag{4.6}$$

El algoritmo del centro de gravedad consume más recursos de hardware de la FPGA, pero obtiene más precisión en la determinación de las coordenadas centrales del objeto, mientras que el *Bounding Box* es muy simple y requiere muchos menos recursos, pero es sensible al ruido que puede afectar a la imagen, como el producido por la mala iluminación o el movimiento del objeto. En esta Tesis se implementan los dos enfoques y se utiliza un conmutador en la FPGA para poder elegir entre los dos métodos en tiempo de ejecución, siempre y cuando se utilice el método de pasada doble de etiquetado de componentes conexas.

4.3 Experimentos

Se muestran a continuación dos experimentos para evaluar el sistema de visión embebido en FPGA propuesto. En el primer experimento se utiliza el método de detección de objetos de doble pasada, mientras que en el segundo experimento se utiliza el algoritmo de pasada simple. Los patrones utilizados tienen objetos cóncavos que presentan problemas en la detección y permiten evaluar la corrección del sistema propuesto. Finalmente, en el Apartado 4.3.3 se evaluará el rendimiento del sistema comparando los resultados de los dos experimentos, así como con el mismo sistema de visión implementado de manera secuencial sobre un PC y con otras implementaciones encontradas en la literatura que tratan un problema similar.

4.3.1 Experimento 1

La Figura 4.20 muestra el patrón utilizado para el primer experimento. En este patrón se observan dos figuras típicas para patrones de sistemas de control visual (el cuadrado y el círculo), y un patrón en forma de U que va a permitir evaluar el correcto comportamiento del sistema de detección de doble pasada.

En este primer experimento se utiliza la implementación del método de dos pasadas para el etiquetado de las componentes conexas descrita en el Apartado 4.2.2.4.1. Se muestran en la Figura 4.21 las distintas imágenes que se obtendrían en cada fase del algoritmo de visión propuesto. Sin embargo, es importante recalcar que en la implementación no se obtienen las imágenes intermedias de cada fase. En la implementación en el cauce segmentado propuesto, cada byte del vector que contiene la información de los niveles de intensidad de los píxeles, es procesado en paralelo, de forma que primero se realiza un redimensionado (que a la vez produce un filtrado de media) de la imagen (ver Figura 4.21.a). Posteriormente cada píxel es binarizado mediante un umbral (75 sobre 255 para este experimento). Esta fase puede verse en la Figura 4.21.b. Para la siguiente fase, en la que se realizará una erosión de la imagen (ver Figura 4.21.c), se necesita almacenar dos filas de la imagen, por lo que se introduce una latencia en el sistema. La siguiente es la primera fase del algoritmo de detección de dos pasadas. Como se puede ver en la Figura 4.21.d, el objeto con forma de U se detecta en esta primera fase como dos objetos distintos (azul y amarillo). En la segunda fase, que requiere que se

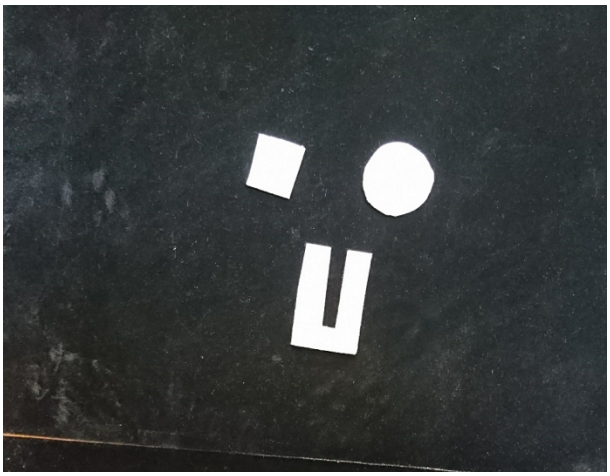


Figura 4.20 Patrón utilizado para el primer experimento: algoritmo de doble pasada

termine la fase anterior y, por lo tanto, es la fase que más latencia introduce en el sistema, se repara el error inicial de la fase 1, detectando ya el objeto en U como un solo objeto (ver Figura 4.21.e). Finalmente, en paralelo con esta segunda fase del algoritmo, se van calculando los centros de masas (ver Figura 4.21.f).

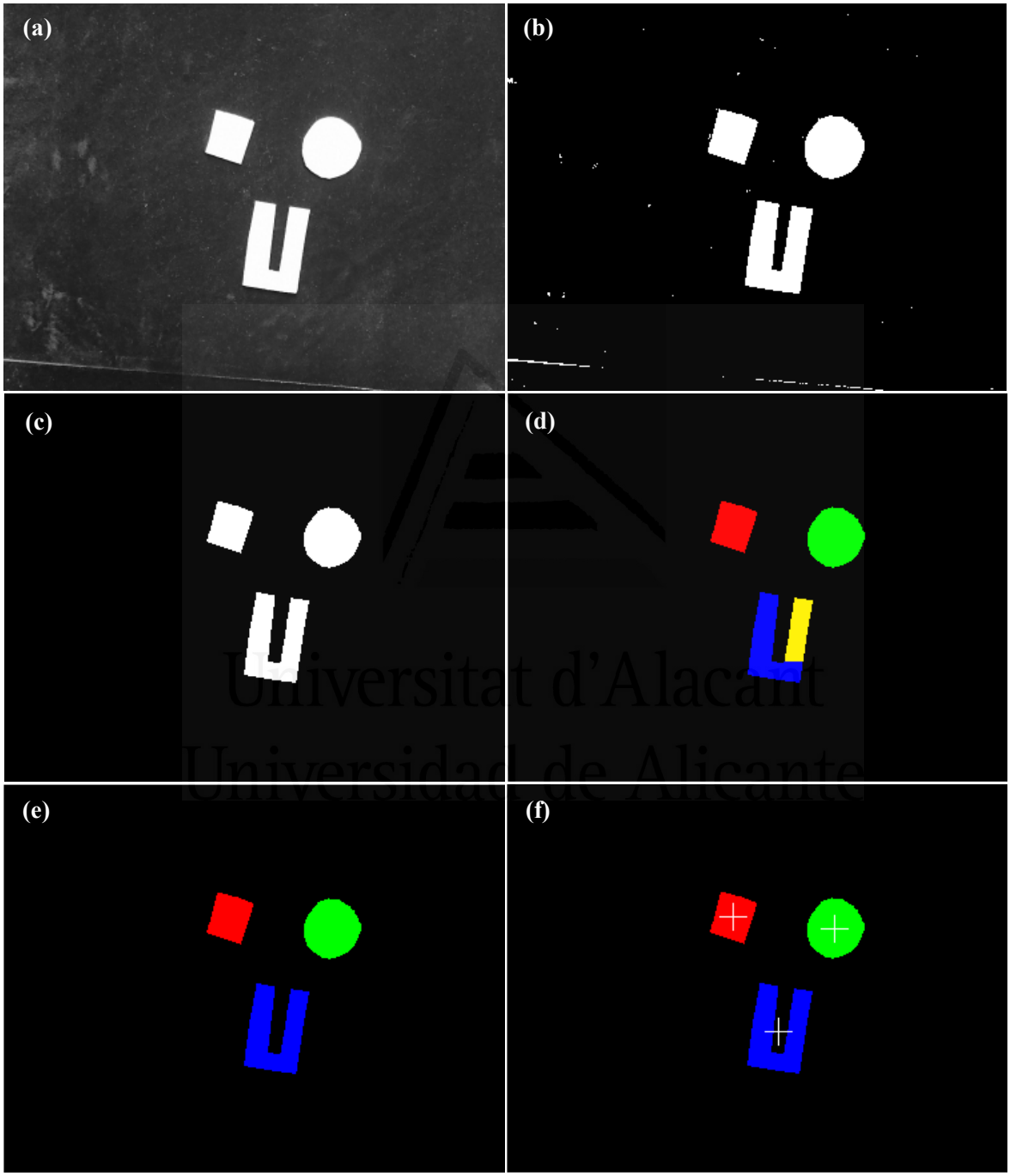


Figura 4.21 Algoritmo de dos pasadas: **(a)** Redimensionado y filtro de media, **(b)** Umbralización, **(c)** Erosión **(d)** Detección de objetos 1ª fase, **(e)** Detección de objetos 2ª fase **(f)** Centroides

La Figura 4.22 muestra el cauce segmentado del sistema para este primer experimento, donde los algoritmos de cada etapa descritos en este Capítulo se han implementado de una forma paralela en la FPGA. Esto representa una de las principales ventajas de implementar algoritmos en una FPGA. La Figura **4.22** muestra también la latencia de cada una de las fases del módulo de visión. Estos módulos se sincronizan con la señal del reloj de píxel de la cámara PCLK. La cámara ha sido configurada para capturar imágenes a máxima resolución (752 x 582 píxeles) y con una tasa de refresco de 340 fps, por lo que el tiempo de adquisición de una imagen completa es de aproximadamente 2.9 ms. El tiempo de exposición depende de las condiciones de luz del entorno y del material de la superficie de la escena. Como se ha descrito anteriormente, en los experimentos se ha empleado un patrón negro con un conjunto de marcas blancas cuyos centroides son las características visuales empleadas por el controlador para realizar el guiado. Con una iluminación menor que la utilizada en los experimentos, el tiempo de exposición tendería a incrementarse. Asimismo, un material plástico o una superficie pulida produciría un incremento del tiempo de exposición. El umbral empleado en la fase de umbralización de este experimento fue de 75. Este umbral es ajustable con un potenciómetro mecánico integrado en la FPGA. La Figura 4.22 muestra que el tiempo de exposición es de 0.1 ms, y la lectura de la imagen completa consume 2.8 ms. En la fase de preprocesado, la resolución de la imagen se reduce a la mitad para sus dos dimensiones. Esto se hace así por dos razones: en primer lugar, para recibir un pixel cada ciclo de reloj y poder mandar dicho píxel al siguiente módulo, y, en segundo lugar, para reducir el ruido en la imagen. Esta reducción de la resolución no implica una reducción del tiempo de ejecución. Sin embargo, esto reduce significantemente el ruido en la imagen ya que actúa como un filtro de suavizado. Además, la memoria necesaria para almacenar la imagen es reducida. Cabe mencionar también que las fases de preprocesado y umbralización se ejecutan simultáneamente según se reciben los píxeles, mientras que la fase de erosión tiene una latencia equivalente al tiempo de envío de dos filas de la imagen. Esto es debido a la naturaleza de esta función morfológica que necesita el valor de intensidad de los píxeles vecinos del píxel actual. Lo mismo ocurre con la primera fase de la etapa de

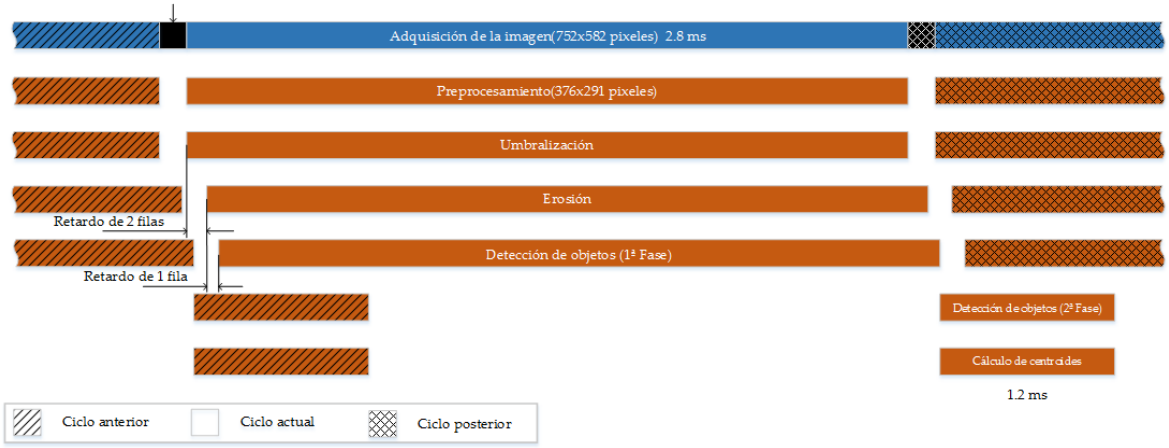


Figura 4.22 Cauce segmentado del módulo de visión embebido en la FPGA para el algoritmo de doble pasada

detección de objetos donde la latencia es de una fila. La segunda etapa de esta fase empieza una vez que la primera ha finalizado, y su latencia es debida a la transferencia de datos entre la FPGA y la memoria. En los experimentos, la latencia es de 1.2 ms usando la memoria interna (Bloques RAM) de la FPGA y el reloj de la tarjeta de 200 MHz. La última fase del módulo de visión determina los centroides de los objetos en paralelo con la fase 2 de la detección de objetos donde se emplea el algoritmo de detección de centros de gravedad. Esto es posible gracias al hecho de que la FPGA dispone de una gran cantidad de recursos hardware. Una vez que se obtienen los centros de gravedad de los objetos, se mandan al módulo de control para que tome las decisiones adecuadas que permitan moverse al robot siguiendo el patrón.

4.3.2 Experimento 2

Para el segundo experimento se ha utilizado un patrón con formas cóncavas más complejas de cara a su etiquetado. La Figura 4.23 muestra el patrón utilizado para este segundo experimento. El objetivo es complicar más al algoritmo de etiquetado para comprobar el correcto funcionamiento del sistema. En este segundo experimento se utiliza la versión final propuesta del sistema de visión, que proporcionará las coordenadas en imagen de las características visuales al controlador. En esta versión final, se utiliza el método de detección de objetos de pasada simple.

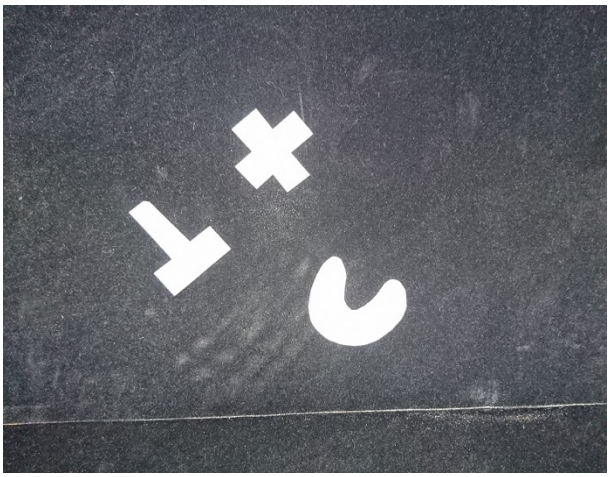


Figura 4.23 Patrón utilizado para el segundo experimento: algoritmo de pasada simple

La Figura 4.24 muestra las 4 fases en que se divide el sistema de visión artificial con pasada simple. A partir de la imagen original, se redimensiona, filtrando al mismo tiempo el posible ruido inicial. Mediante el umbralizado se intenta segmentar los objetos del fondo. Dado que es habitual que quede ruido, la tercera fase de erosión se encarga de eliminar este ruido. Finalmente, la última etapa realiza el etiquetado de los objetos encontrados y calcula su centro de masas.

La Figura 4.25 muestra el cauce segmentado para el método de detección de pasada simple descrito en el Apartado 4.2.2.4.2. Como ocurría con el método de pasada doble, cada módulo del sistema de visión se sincroniza con la señal del reloj de píxel de la cámara PCLK. Para poder comparar tiempos y resultados, la cámara ha sido configurada también para capturar imágenes a máxima resolución (752 x 582 píxeles) y con una tasa de refresco de 340 fps, por lo que, de nuevo, el tiempo de adquisición de una imagen completa es de aproximadamente 2.9 ms. En la primera fase de redimensionado de la imagen se tiene una latencia de 4 ciclos de reloj PCLK. Con esta operación, al igual que ocurría anteriormente, se obtiene un filtrado de media que suaviza el ruido de la imagen capturada por la cámara. La fase de binarizado tiene esta misma latencia, ya que trabaja con cada píxel que proporciona el módulo anterior. Las dos fases, como se puede observar en el cronograma de la Figura 4.25 se realizan en paralelo. Para la siguiente etapa se requiere almacenar dos filas de píxeles previas. Esto se realiza en memorias intermedias. Por último, la etapa de detección de objetos y cálculo de centroides requiere de una fila

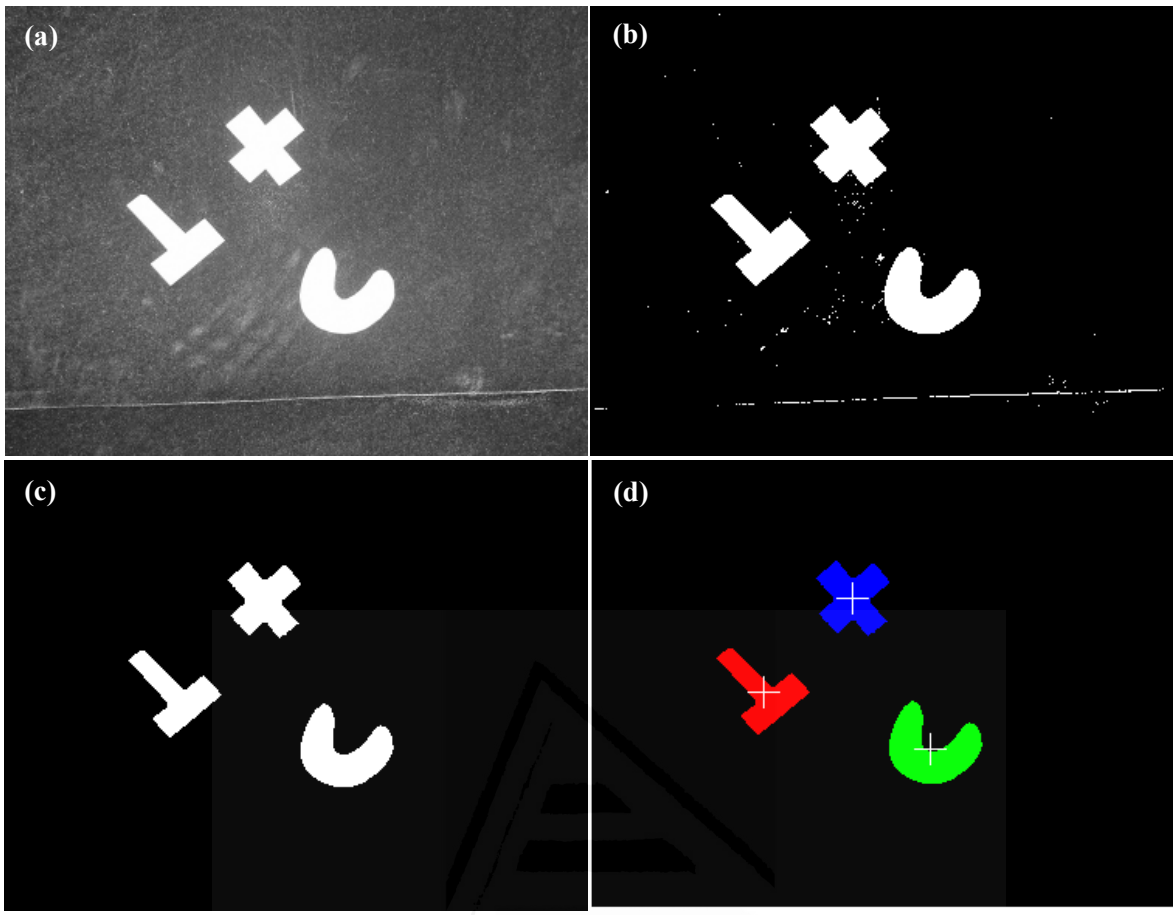


Figura 4.24 Algoritmo de pasada simple: **(a)** Redimensionado y filtro de media, **(b)** Umbralización, **(c)** Erosión **(d)** Detección de objetos y cálculo de centroides

previa de la imagen, por eso tiene esa latencia de una fila que muestra la Figura 4.25. Otro aspecto a resaltar en cuanto a esta última fase es que el algoritmo de etiquetado está configurado con un límite inferior y superior de objetos. Esto es debido a que, en control visual, se trabaja con 1 punto característico cuando se quiere realizar posicionamientos en un plano y se tienen al menos dos grados de libertad en el robot. Cuando se trabaja en el espacio Cartesiano, con orientación y translación del extremo del robot, y con un robot de al menos 6 grados de libertad, los patrones utilizados para la extracción de características visuales son normalmente de 4 puntos. Aunque es posible trabajar con más puntos característicos, con 4 puntos es posible asegurar la estabilidad de los controladores visuales. Configurado de esta forma, el cálculo de los centroides para este método de pasada simple se reduce a la división de dos de los valores guardados en la LUT que

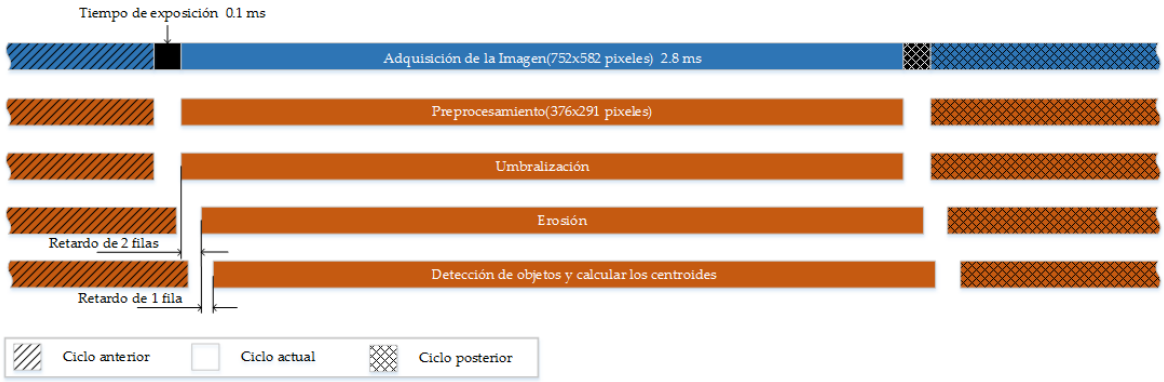


Figura 4.25 Cauce segmentado del módulo de visión embebido en la FPGA para el algoritmo de pasada simple

almacenan las etiquetas de los objetos detectados (ver Figura 4.19). Dado que mediante un parámetro se acota el valor máximo de objetos que se desean detectar, el número de divisiones está acotado, siendo la latencia de este módulo de 57 ns más 1 ns por cada objeto adicional encontrado. Así, el sistema obtiene una respuesta nueva (centroides de los objetos encontrados) en un tiempo fijo y constante.

4.3.3 Rendimiento del sistema de visión artificial propuesto

Para evaluar el comportamiento del sistema propuesto, se implementa el mismo sistema de visión artificial sobre PC. Sobre un PC el sistema de visión sería un sistema secuencial, en el que las imágenes procesadas se irían almacenando en RAM. Si se implementan exactamente las mismas operaciones que se han implementado en la arquitectura semiabierta propuesta en esta Tesis para la detección de centroides de los objetos visuales sobre la imagen, se empezará con una operación de redimensionado de la imagen, que al mismo tiempo permite realizar un filtrado de media. Posteriormente, se umbraliza la imagen para luego pasar a una etapa de erosión. La siguiente fase permitiría detectar las componentes conexas y por último se calcularía los centros de masas de estos objetos encontrados en la imagen.

Para los experimentos se ha utilizado un PC con un procesador de 64 bits Intel Core i5-4400 a 3.1 GHz, con 8 GB de RAM. La Tabla 4.1 muestra 10 experimentos distintos sobre el mismo patrón mostrado en la Figura 4.23. Como puede observarse, los tiempos entre

Tabla 4.1 Tiempos de cada fase en distintos experimentos sobre el algoritmo de visión implementado en un PC

	Filtrado (ms)	Umbral (ms)	Erosión (ms)	Detección (ms)	Centroides (ms)	Tiempo total (ms)
Prueba #1	8.154	0.299	0.996	0.578	1.554	11.581
Prueba #2	7.59	0.264	0.916	0.35	1.408	10.528
Prueba #3	7.736	0.238	0.738	0.374	1.211	10.297
Prueba #4	7.573	0.229	0.799	0.322	1.856	10.779
Prueba #5	7.659	0.29	0.863	0.318	1.317	10.447
Prueba #6	7.601	0.237	0.787	0.318	1.296	10.239
Prueba #7	7.604	0.256	0.733	0.324	1.478	10.395
Prueba #8	7.589	0.244	0.797	0.754	2.418	11.802
Prueba #9	7.554	0.242	0.817	0.344	1.451	10.408
Prueba #10	7.686	0.235	0.758	0.335	1.514	10.528
Media	**7.6746**	**0.2534**	**0.8204**	**0.4017**	**1.5503**	**10.7004**

experimentos no son constantes. Esto es debido a que el sistema operativo no es un sistema operativo en tiempo real. El procesador debe, por tanto, atender a otros procesos, con lo que las operaciones, aun siendo las mismas, pueden no tardar el mismo tiempo.

Si a los tiempos medios que se indican en la Tabla 4.1 se añade el tiempo de captura y envío de una imagen hacia el ordenador, se obtiene el esquema de tiempos medios que puede verse en la Figura 4.26. Es importante recalcar el hecho de que el esquema muestra la media de los 10 experimentos realizados. Este tiempo de 13.6 ms no será constante, con lo que el sistema de control no tendrá la posición en imagen de las características visuales a un ritmo constante, lo que impide que el sistema se comporte como un sistema de tiempo real.

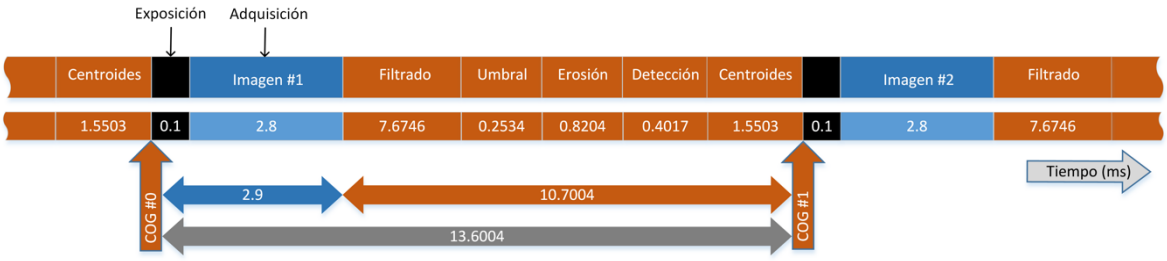


Figura 4.26 Tiempo medio entre dos medidas del centro de masas de las características visuales en una implementación secuencial sobre PC

La propuesta realizada en este Capítulo de la Tesis para el sistema de visión, capaz de proporcionar características visuales a un controlador visual directo, obtiene como se ha comentado una respuesta en un tiempo fijo estable. Esto es posible gracias al diseño en paralelo del algoritmo, que permite que cada fase se pueda procesar mientras se están procesando las otras fases. El principal cuello de botella del sistema propuesto es la tasa de refresco de imágenes de la cámara. Algo que, por otro lado, queda fuera del funcionamiento de la propuesta realizada. La Tabla 4.2 muestra los tiempos de cada uno de los métodos implementados sobre la FPGA. El método 1 se refiere al método en que la detección de las componentes conexas se realiza con el algoritmo de dos pasadas, mientras que el método 2 se refiere a la implementación con el algoritmo de pasada simple. Esta última implementación es la que se utilizará finalmente para proporcionar la posición actual de las características visuales al controlador que se describirá en el siguiente capítulo. Para este segundo método, el tiempo de proceso de un fotograma es de sólo 28.9 us, después de haber recibido el último píxel de la imagen a través del flujo de datos que se establece con el digitalizador de vídeo. Esto permitiría trabajar con hasta 34558 fotogramas por segundo, siempre y cuando se tuviera una cámara capaz de trabajar a esa frecuencia. La cámara utilizada en los experimentos realizados a lo largo de esta Tesis, tarda 2.9 ms en proporcionar una imagen completa, desde que se realiza la exposición de la imagen en el CCD (para lo que se invierten 0.1 ms), hasta que envía a la FPGA todo el vector de bytes que conforma la imagen de 752 x 582 píxeles. Por lo tanto, dado que el tiempo de captura de imagen es del orden de 1000 veces mayor que el tiempo de proceso requerido una vez se ha concluido de enviar la imagen, queda claro que el

Tabla 4.2 Tiempo constante en experimentos con la FPGA. Método 1: Propuesta de sistema de visión artificial con detección de componentes conexas de doble pasada; Método 2: Propuesta de sistema de visión artificial con detección de componentes conexas de pasada simple

	Filtrado (ns)	Umbral (ns)	Erosión (us)	Detección (ms)	Centroides (ns)	Tiempo total (ms)
Método 1	26.8	26.8	19.2	1.2096	58	1.2289116
Método 2	26.8	26.8	19.2	0.0096	58	0.0289116

cuello de botella del sistema es la captura de imagen. Si se utilizara el método de doble pasada para el etiquetado de los objetos, el tiempo de captura sería sólo poco más del doble que el tiempo de proceso final necesario para obtener los centroides de los objetos detectados. Este método es por tanto mucho menos eficiente que el método de pasada simple propuesto. Otra conclusión que se puede extraer al visualizar la Tabla 4.1 y la Tabla 4.2, es que los tiempos en PC son mucho mayores que los tiempos empleados por la FPGA para las mismas operaciones. Esto se debe a que la FPGA emplea hardware dedicado exclusivamente a cada operación, trabajando además de manera paralela, permitiendo así que el tiempo total del proceso sea mucho menor.

En la Figura 4.27 se puede ver que el tiempo necesario para terminar el proceso en el método de dos pasadas es relevante comparándolo con el tiempo de captura. También se aprecia que, aun así, el tiempo entre el cálculo de las posiciones de los objetos en una iteración y la siguiente siempre será estable y de 2.9 ms (que se corresponde con el tiempo de captura de la imagen).

Finalmente, la Figura 4.28 muestra el esquema de tiempos entre dos fotogramas para el método de pasada simple, donde se observa que el tiempo de proceso requerido tras obtener el último píxel de la imagen hasta que se obtienen los centroides es de sólo 28 us. Este tiempo es prácticamente despreciable frente a los 2.9 ms que tarda la cámara en proporcionar una imagen. De nuevo, esto no influye en que el periodo estable en el que el sistema de visión puede proporcionar un nuevo valor de los centros de masas

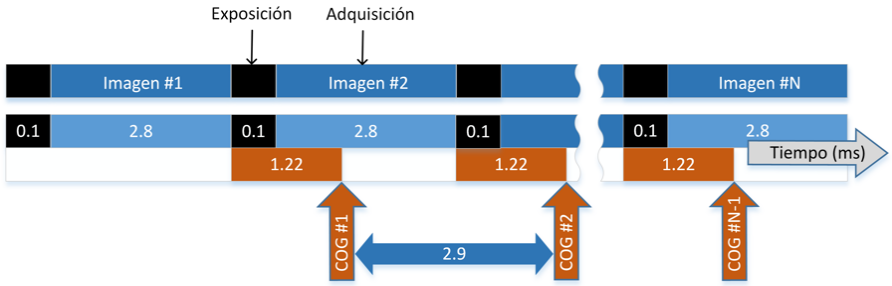


Figura 4.27 Tiempo constante entre dos medidas del centro de masas de las características visuales en la implementación paralela empleando el método de detección de componentes conexas de dos pasadas

de los objetos detectados es de 2.9 ms. Sin embargo, en el resto de experimentos realizados en esta Tesis, se empleará a partir de ahora el método de detección de componentes conexas con pasada simple, ya que, aunque no afecta en el tiempo entre muestras a proporcionar al controlador, sí que es más eficiente en recursos y mejora enormemente los tiempos de procesamiento de la imagen, con lo que la respuesta se obtiene en un tiempo más cercano a cuando se ha capturado la imagen.

Para terminar con la evaluación del rendimiento del sistema de visión propuesto, es importante compararlo con otras implementaciones en las que el objetivo del sistema de visión es similar, es decir, la obtención del centro de masas de los objetos que se detecten en la imagen. Para ello, la Tabla 4.3 resume las especificaciones de una serie de implementaciones de sistemas de visión artificial sobre FPGA para la detección de objetos. Desafortunadamente, no todos los autores aportan todos sus parámetros de consumo de

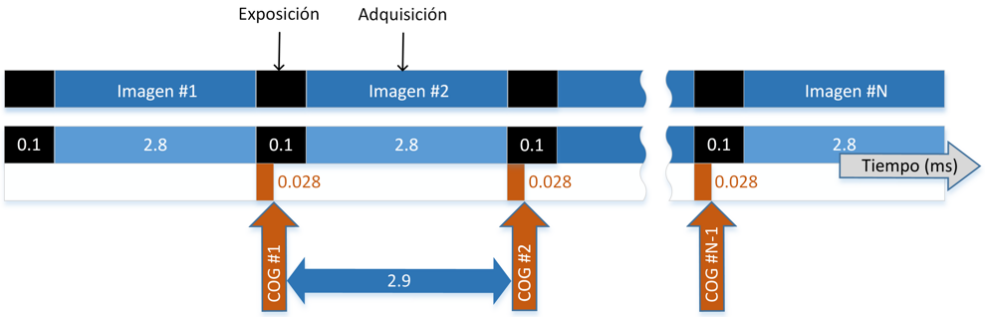


Figura 4.28 Tiempo constante entre dos medidas del centro de masas de las características visuales en la implementación paralela empleando el método de detección de componentes conexas de pasada simple

Tabla 4.3 Comparativa de algoritmos de detección de objetos visuales sobre FPGA (Fuente: [Acevedo-Avila-2016])

Algoritmo	Plataforma	Fotograma	Píxel (bit)	Memoria (bits)	Frecuencia (MHz)	fps
[Calvo-Gallego-2012]	Xilinx Spartan-3A	640 × 480	8	2,462,720	27	60
[Kiran-2013] (1)	Xilinx Virtex V	100 × 100	8	144,000	100	4545
[Kiran-2013] (2)	Xilinx Virtex V	1024 × 1024	8	-	100	61
[Mauch-2014]	Xilinx Spartan-6	224 × 224	12/10	162,000	70	905
[Klaiber-2013]	Xilinx Virtex VI	1024 × 1024	-	1,512,000	136.4	1049
[Klaiber-2016]	Xilinx Kintex 7	7680 × 4330	-	548,000	170.3	-
[Yuhai-2011]	Altera Stratix II	1280 × 1280	1	75,600	97.4	49
[Bochem-2011]	Altera DE2 Cyclone II	640 × 480	-	239,316	125	50
[Acevedo-Avila-2016]	Altera Cyclone III	640 × 480	1	307,200	50	110

recursos. Además, no se utiliza el mismo tamaño de fotograma en todas las arquitecturas, con lo que es difícil compararlos en igualdad de condiciones.

Para poder comparar los métodos que se muestran en la Tabla 4.3 con el sistema de visión propuesto en este Capítulo de esta Tesis, la Tabla 4.4 resume los mismos parámetros de rendimiento. La propuesta realizada en esta Tesis reduce al mínimo el consumo de memoria y cumple con los requisitos de tiempo real que beneficiarán tanto al controlador visual que se propondrá en el siguiente Capítulo. Sobre todo, con el método de pasada simple, que permite procesar hasta 34558 fotogramas por segundo si la cámara de captura del sistema fuera capaz de obtener imágenes a esa velocidad. El resultado más cercano es más de 34 veces más lento [Klaiber-2013], con una imagen, eso sí, de mayor tamaño. También es importante reseñar que el sistema de visión propuesto con pasada

Tabla 4.4 Comparativa de algoritmos de detección de objetos visuales sobre FPGA: Métodos propuestos en esta Tesis

Algoritmo	Plataforma	Fotograma	Píxel (bit)	Memoria (bits)	Frecuencia (MHz)	fps
Sistema de visión propuesto (dos pasadas)	Xilinx Kintex 7	752 × 582	8	878,336	200	813
Sistema de visión propuesto (pasada simple)	Xilinx Kintex 7	752 × 582	8	3008	200	34558

simple es el que menos memoria de la FPGA utiliza para realizar la detección, también del orden de 25 veces menos que el método que menos utiliza de los mostrados en la Tabla 4.3.

En la Tabla 4.5 se muestran otros algoritmos propuestos para la detección de objetos, en este caso sobre GPPs, GPUs o DSPs. La mayoría obtienen un rendimiento cercano al refresco de las cámaras estándar de 30 fps. Sin embargo, el alto consumo de recursos dificulta su implementación sobre hardware embebido, y los tiempos no mejoran a los tiempos de los algoritmos embebidos en FPGA, quedándose en algunos casos muy lejos de los tiempos constantes proporcionados por el sistema de visión propuesto en esta

Tabla 4.5 Comparativa de algoritmos de detección de objetos visuales sobre GPPs, GPUs o DSPs (Fuente: [Acevedo-Avila-2016])

Algoritmo	Tecnología	Fotograma	Memoria	Tiempo proceso	fps
[Paralic-2012] (1)	Intel Atom N280	320 × 240	1 GB	1.13 ms	30
[Paralic-2012] (2)	Intel Atom N280	640 × 480	1 GB	4.86 ms	30
[NguyenTB-2009]	Pentium 4 (3 GHz)	320 × 240	3 GB	3.72 ms	N/A
[Swati-2014]	Intel Core 2 Duo	800 × 1200	3 GB	21 ms	N/A
[Oro-2011]	NVidia GTX 470 GPU	1920 × 1080	1.2 GB	2.3 ms	35
[KumarV-2015]	TriMedia DSP	640 × 480	N/A	4.2 ms	30

Tesis. A los tiempos mostrados en esta tabla hay que sumar el tiempo de adquisición y almacenamiento en memoria de cada fotograma.

4.4 Conclusiones

El sistema de visión de un sistema de control visual es el responsable de proporcionar información de realimentación visual al controlador del robot para calcular la acción de control apropiada. En esta sección se ha descrito una arquitectura parcialmente abierta embebida en una FPGA, que es capaz de proporcionar la información visual mucho más rápida que la proporcionada por un sistema desarrollado en un GPP como sería, por ejemplo, un ordenador personal.

La propuesta realizada en este Capítulo es un método capaz de aprovechar al máximo los recursos que proporciona una FPGA. Para ello se propone el diseño de un cauce segmentado, que permite ejecutar en tiempo real el flujo de datos proporcionado por el digitalizador de vídeo. Así, la FPGA es capaz de procesar cada píxel para, por un lado, segmentar el fondo de los posibles objetos en la imagen y, por otro lado, etiquetar los objetos encontrados y obtener su centroide. El cauce segmentado permite trabajar a la frecuencia máxima de la cámara. El tiempo de respuesta está por lo tanto acotado, lo que convierte al sistema propuesto en un sistema de tiempo real, aspecto éste que se aprovechará para proporcionar características de tiempo real al controlador visual directo propuesto en esta Tesis. El tratamiento de la imagen como un flujo continuo de datos en el cauce segmentado proporciona la ventaja de no requerir más que dos líneas de la imagen en memoria intermedia, reduciendo así al máximo la necesidad de uso de memoria en la FPGA.

El sistema de visión artificial propuesto permite el cálculo de los centroides de los objetos detectados en cada fotograma de un vídeo. Para la detección y etiquetado de estos objetos, se han implementado dos algoritmos distintos. En un principio se optó por el método de dos pasadas. El algoritmo de pasada simple permite mejorar no sólo los tiempos proporcionados por el método de dos pasadas, sino también los recursos utilizados por este método previo. Los experimentos demuestran que el cuello de botella del sistema propuesto es la propia cámara, que a pesar de trabajar a una muy alta

velocidad de 340 fps, no es capaz de trabajar a la velocidad de procesamiento a la que se ha llegado con las optimizaciones propuestas. La comparativa del sistema embebido con un algoritmo similar programado sobre un PC recalca el valor del diseño propuesto. Además, comparando el método propuesto con otros métodos de detección de objetos y cálculo de su centro de masas, se observa que los esfuerzos de optimización han logrado resultados mucho mejores tanto en número de fotogramas por segundo que el sistema podría procesar (entendiendo aquí que se pudiera disponer de una cámara mucho más rápida), como en la reducción de recursos utilizados. La propuesta realizada con el algoritmo de pasada simple sería capaz de alcanzar los 34558 fps si fuera posible trabajar con una cámara que proporcionara las imágenes a esa tasa de refresco. Y todo ello trabajando en tiempo real.

En el siguiente capítulo se describirá el módulo de control de la arquitectura embebida propuesta en esta Tesis. Este módulo tiene como entrada las coordenadas en imagen de los centros de masas calculados por el módulo de visión. A partir de esta información, y de las coordenadas de los centros de masas de esos mismos objetos en la posición final deseada, el controlador calcula en cada iteración del bucle de control los pares articulares que moverán al extremo del robot (y junto a él a la cámara en él situada) hasta su objetivo.

5 Control visual directo basado en FPGA

Los sistemas de control visual de robots manipuladores permiten el posicionamiento del extremo del robot utilizando información visual de su entorno. En concreto, un sistema de control visual directo, proporciona los pares articulares capaces de llevar el extremo del robot a la posición deseada. Es evidente que cualquier mejora en los tiempos de procesamiento de las imágenes capturadas por la cámara que guía al robot, mejorará el rendimiento general del sistema. Los esfuerzos realizados para obtener un controlador más rápido y preciso al diseñar controladores directos en lugar de controladores cinemáticos indirectos, pierden sentido si no se optimiza cada proceso de recogida de información y cálculo de operaciones dentro del controlador.

En este Capítulo se describen las optimizaciones acerca del controlador llevadas a cabo en la arquitectura semiabierta propuesta en el Capítulo 3. En el Capítulo 4 se describieron las ventajas obtenidas al implementar sobre una FPGA el sistema de visión encargado de proporcionar posiciones actualizadas de las características visuales. En este capítulo se describe la implementación de distintos controladores visuales directos, desde el clásico de Jacobiana transpuesta, pasando por el de dinámica inversa, hasta llegar a la implementación de un framework que permite desarrollar nuevos controladores visuales directos en función de una matriz **W**.

Para validar los distintos controladores implementados sobre la FPGA y poder comparar sus comportamientos con respecto a sistemas implementados sobre GPPs, se emplean dos robots manipuladores: COOPER, un robot planar de 3 gdl; y el Mitsubishi PA10, un robot antropomórfico de 7 gdl.

5.1 Introducción

Como se ha descrito a lo largo de la Tesis, el uso de FPGAs permite disponer de una tecnología hardware reprogramable adecuada para la implementación de controladores visuales. Los sistemas de control visual clásicos permiten realizar el posicionamiento punto a punto de un robot haciendo uso de información visual [Chaumette-2006]. En

contraste con las aproximaciones clásicas, los sistemas de control visual directos o dinámicos realizan el control articular directo del robot mandando aceleraciones articulares o pares al robot a partir de la información visual capturada. Como se describirá a lo largo del presente capítulo, la implementación de un sistema de control visual directo totalmente embebido en una FPGA permite mejorar el comportamiento de estos sistemas, reduciendo los tiempos de procesamiento y obteniendo retardos estables en la realimentación. Por lo tanto, el uso de una FPGA permite la implementación de una arquitectura paralela dedicada que puede ser adaptada en tiempo real a las necesidades específicas que tenga el sistema.

Tal y como se ha descrito en el Capítulo 2, las FPGAs se han utilizado en diferentes sistemas de control que requieren la implementación de controladores lógicos difusos [Sulaiman-2009], sistemas de control multieje [Astarloa-2009], redes neuronales [Muthuramalingam-2008], sistemas mecatrónicos [MacCleery-2008], etc. Dentro del campo de los sistemas de control de robots, es conveniente mencionar trabajos previos como los descritos en [Lalpuriva-2013], que emplean FPGAs para el diseño de controladores para robots debido a su paralelismo y a la flexibilidad que permiten. De esta manera, es posible encontrar arquitecturas para el control de robots que integran FPGAs en alguno de los componentes de su diseño [Shao-2006]. En este último trabajo, los algoritmos son particionados en una parte lineal, que es implementada en una FPGA, y una porción no lineal, que es implementada en un DSP. Por ejemplo, en [ChenZP-2009] se presenta un sistema de control basado en FPGA para el control de impedancia de una mano robótica. El trabajo presentado en [ZhangL-2012] describe una arquitectura de control basada en FPGA que puede ser fácilmente aplicada a cualquier robot. Podemos encontrar otro ejemplo en [LeeWK-2006], en el que se describe un controlador basado en FPGA para el brazo de un robot humanoide que implementa varias tareas, como controladores PID, comunicación, contadores de encoder o generadores PWM (del inglés *Pulse Width Modulation*). En [Wen-Hong-2013] se propone un controlador embebido basado en FPGA para su aplicación a robots modulares. Sin embargo, a pesar de que las FPGAs están cada vez más integradas en los sistemas de control de un gran número de aplicaciones, no existen implementaciones previas de arquitecturas para el control visual

directo de robots manipuladores basadas en FPGAs. Dentro del ámbito de los sistemas de control visual, hay pocas implementaciones que integran FPGAs en algún elemento del sistema de control visual. La mayoría de los retardos en estos sistemas son debidos a la necesidad de procesamiento de imágenes. Este es el motivo por el que hay algunos trabajos como [Liyanage-2014] que optimizan el proceso de captura de imágenes a través de una implementación hardware en FPGA. Adicionalmente, es posible encontrar controladores visuales cinemáticos (o indirectos) que integran procesamiento de imagen, así como control [Jwu-Sheng-2011]. En [Tu-2011] se describe un sistema de control para realizar el control visual robusto de un péndulo invertido empleando un co-procesador basado en imagen, y [ParkJH-2009] presenta un sistema de control visual que consiste en un sistema de procesamiento de imágenes y un robot industrial para mover una superficie soportada por el robot a lo largo de una trayectoria. Sin embargo, no es posible encontrar arquitecturas basadas en FPGAs para implementar controladores visuales directos con el objetivo de realizar el guiado de robots manipuladores empleando información visual. En el presente capítulo se formulará un nuevo sistema de control visual directo basado en imagen para el seguimiento de trayectorias. Este sistema de control ha sido implementado empleando la arquitectura propuesta basada en FPGA descrita en el Capítulo 3. A lo largo del capítulo se describirán los detalles de su implementación, así como los resultados obtenidos. Se comparará el controlador propuesto con una implementación en FPGA de un sistema de control visual directo basado en Jacobiana transpuesta. También se implementará en la FPGA un controlador basado en control óptimo cuya precisión es ajustable dependiendo de la función de peso utilizada.

Como se describe en un trabajo previo [Pomares-2014], cuando se aplica un controlador visual es posible obtener un comportamiento articular caótico durante el seguimiento de una trayectoria repetitiva por parte del extremo del robot. Cuando se obtiene un comportamiento caótico, se produce un movimiento articular no periódico e impredecible durante el seguimiento de la trayectoria deseada en el espacio imagen. Con el objetivo de compensar el comportamiento articular caótico, se propone utilizar el método descrito en [Pomares-2014], que hace uso de un controlador con realimentación retardada DFC (del inglés *delayed feedback controller* [Pyragas-2006]). Para ajustar la

ganancia de este controlador se usan varias simulaciones que determinan la relación entre la ganancia y el exponente máximo de Lyapunov del sistema. Este método supone que se obtiene un movimiento articular periódico usando una ganancia con la cual el exponente máximo de Lyapunov del sistema es negativo. El controlador visual directo propuesto se extenderá para integrar esta compensación de caos basada en FPGA. Como se demuestra con los resultados experimentales, el uso de este método con la arquitectura basada en FPGA permite compensar mejor los comportamientos caóticos que se obtienen, debido a que los retardos son reducidos además que los tiempos de procesamiento se mantienen constantes. Las principales aportaciones del sistema de control visual propuesto basado en FPGA se han publicado en [Alabdo-2016a].

Tras esta introducción, el capítulo presenta un segundo apartado 5.2 en el que se describe la formulación del controlador visual directo propuesto basado en imagen para el seguimiento de trayectorias. En el apartado 5.3 se describe cómo se ha realizado la implementación del controlador propuesto en una FPGA. También se describe la implementación en la misma FPGA de un controlador basado en Jacobiana transpuesta, así como la implementación de un controlador visual directo basado en control óptimo. En el apartado 5.4 se extiende el controlador con el objetivo de integrar el compensador del comportamiento caótico. En el apartado 5.5 se describen detalles de las implementaciones del controlador propuesto empleando dos robots: robot COOPER de 3 grados de libertad y el robot Mitsubishi PA-10 con 7 grados de libertad. Finaliza el capítulo con un resumen de las principales conclusiones y aportaciones.

5.2 Control visual directo

En primer lugar, se van a definir algunas cuestiones relativas a la notación del controlador empleado para el guiado del robot. Se considera $\mathbf{q} \in \Re^{nx1}$ las coordenadas articulares del robot, $\dot{\mathbf{q}} \in \Re^{nx1}$ representan las velocidades articulares mientras que $\ddot{\mathbf{q}} \in \Re^{nx1}$ son las aceleraciones articulares (n representa los grados de libertad o número de articulaciones del robot). Como se ha indicado anteriormente, el robot es guiado por información visual. En un caso general, esta información visual es un vector de k puntos

en el espacio imagen, que se representan con el vector $\mathbf{s} = \left[f_{1x}, f_{1y}, f_{2x}, f_{2y}, \ldots, f_{kx}, f_{ky}\right]^T \in \Re^{2k}$. Según se describió en el Capítulo 2, la matriz de interacción establece la siguiente relación entre el espacio imagen y el espacio Cartesiano 3D:

$$\dot{\mathbf{s}} = \mathbf{L}_s(\mathbf{s}, \mathbf{Z})\dot{\mathbf{r}} \tag{5.1}$$

donde $\dot{\mathbf{s}}$ es la derivada con respecto al tiempo de las características visuales extraídas, $\mathbf{r}$ representa la localización 3D del sistema de coordenadas asociado a la cámara del extremo del robot y $\dot{\mathbf{r}}$ es la velocidad correspondiente a la cámara. Se va a considerar que el robot se encuentra realizando el seguimiento en un espacio m-dimensional (por lo tanto, $\mathbf{r} \in \Re^m$) de forma que si n > m el robot será redundante. Dado que se considera un control visual con cámara montada en el extremo o eye-in-hand, la relación entre la cámara y el extremo del robot es fija y conocida. De ahí que se pueda hablar de $\mathbf{r}$ o $\mathbf{r}_e$ indistintamente.

Por otro lado, la Jacobiana del robot relaciona la velocidad del efector final del robot y su velocidad articular, $\dot{\mathbf{q}} : \dot{\mathbf{r}} = \mathbf{J}_r(\mathbf{q})\dot{\mathbf{q}}$

$$\dot{\mathbf{r}} = \mathbf{J}_r(\mathbf{q})\dot{\mathbf{q}} \tag{5.2}$$

Haciendo uso de la Ecuaciones (5.1) y (5.2) es posible obtener la siguiente ecuación que representa la relación entre la velocidad articular del robot y la variación con respecto al tiempo de las características visuales:

$$\dot{\mathbf{s}} = \mathbf{L}_s(\mathbf{s}, \mathbf{Z})\mathbf{J}_r(\mathbf{q})\dot{\mathbf{q}} = \mathbf{L}_J(\mathbf{q}, \mathbf{s}, \mathbf{Z})\dot{\mathbf{q}} \tag{5.3}$$

donde $\mathbf{L}_J = \mathbf{L}_s(\mathbf{r})\mathbf{J}_r(\mathbf{q}) = \mathbf{L}_J(\mathbf{q}, \mathbf{s}, \mathbf{Z}) \in \Re^{2k \times n}$. Derivando la Ecuación (5.3) con respecto al tiempo se puede obtener la siguiente expresión (con el objetivo de obtener una notación más compacta en las siguientes ecuaciones no se indica la dependencia de las distintas Jacobianas con respecto a las variables articulares):

$$\ddot{\mathbf{s}} = \mathbf{L}_J\ddot{\mathbf{q}}_r + \dot{\mathbf{L}}_J\dot{\mathbf{q}} \tag{5.4}$$

donde $\ddot{\mathbf{s}}$ representa la aceleración en el espacio imagen o segunda derivada con respecto al tiempo de las características visuales extraídas. Con el objetivo de permitir realizar el correcto seguimiento de una trayectoria especificada en el espacio imagen se va a

proponer un controlador basado en aceleración en el espacio articular. En la bibliografía a menudo es posible encontrar aproximaciones basadas en velocidad debido a su simplicidad. Sin embargo, la aproximación basada en aceleración es más adecuada cuando se tiene en cuenta la dinámica del robot empleado en el control. Así, con $\ddot{\mathbf{q}}_r$ se representa la aceleración articular que será aplicada al robot y generada por el controlador con el objetivo anteriormente comentado.

La acción de control o aceleraciones articulares requeridas, $\ddot{\mathbf{q}}_r$, pueden ser obtenidas a partir de (5.4):

$$\ddot{\mathbf{q}}_r = \mathbf{L}_J^+\left(\ddot{\mathbf{s}}_r - \dot{\mathbf{L}}_J\dot{\mathbf{q}}\right) \tag{5.5}$$

Queda por definir la referencia en el espacio imagen, $\ddot{\mathbf{s}}_r$, que permita desarrollar correctamente el seguimiento de las características visuales. Para ello se considera:

$$\ddot{\mathbf{s}}_r = \ddot{\mathbf{s}}_d + \mathbf{K}_D\left(\dot{\mathbf{s}}_d - \dot{\mathbf{s}}\right) + \mathbf{K}_P\left(\mathbf{s}_d - \mathbf{s}\right) = \ddot{\mathbf{s}}_d + \mathbf{K}_D\dot{\mathbf{e}}_s + \mathbf{K}_P\mathbf{e}_s \tag{5.6}$$

donde $\ddot{\mathbf{s}}_d$, $\dot{\mathbf{s}}_d$ y $\mathbf{s}_d$ son las aceleraciones, velocidades y posiciones deseadas para las características en la imagen. $\mathbf{K}_P$ y $\mathbf{K}_D$ son matrices PD de ganancias. Además, $\mathbf{e}_s$, es el error en la imagen, y $\dot{\mathbf{e}}_s$ es la derivada con respecto al tiempo del error en el espacio imagen.

A continuación, se extenderá el sistema de control visual para generar directamente los pares articulares necesarios para realizar el seguimiento. Para ello, se parte del modelo dinámico del robot:

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}}_r + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) \tag{5.7}$$

donde $\mathbf{M}(\mathbf{q}) \in \Re^{nxn}$ es la matriz de inercia del manipulador (definida positiva y simétrica), $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) \in \Re^{nx1}$ es el vector de los pares centrípetos y de Coriolis, $\mathbf{g}(\mathbf{q})$ es la fuerza gravitacional. Finalmente, $\boldsymbol{\tau} \in \Re^{nx1}$ es el vector de los pares aplicados a las articulaciones. A partir de la utilización de las Ecuaciones (5.5), (5.6) y (5.7), el controlador visual directo que se obtiene será el que se representa en la siguiente ecuación:

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\mathbf{L}_J^+\left(\ddot{\mathbf{s}}_d + \mathbf{K}_D\dot{\mathbf{e}}_s + \mathbf{K}_P\mathbf{e} - \dot{\mathbf{L}}_J\dot{\mathbf{q}}\right) + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) \tag{5.8}$$

Teniendo en cuenta las hipótesis previas, la obtención del lazo cerrado se puede obtener de la siguiente manera:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}}_r = \mathbf{M}(\mathbf{q})\mathbf{L}_J^{+}\left(\ddot{\mathbf{s}}_d + \mathbf{K}_D\dot{\mathbf{e}}_s + \mathbf{K}_P\mathbf{e} - \dot{\mathbf{L}}_J\dot{\mathbf{q}}\right) \tag{5.9}$$

Realizando la pre-multiplicación de $\mathbf{L}_J\left(\mathbf{M}(\mathbf{q})\right)^{-1}$ en ambos lados de la ecuación (5.9):

$$\mathbf{L}_J\ddot{\mathbf{q}}_r = \ddot{\mathbf{s}}_r - \dot{\mathbf{L}}_J\dot{\mathbf{q}} \rightarrow \mathbf{L}_J\ddot{\mathbf{q}}_r + \dot{\mathbf{L}}_J\dot{\mathbf{q}} = \ddot{\mathbf{s}}_r \tag{5.10}$$

Tomando en consideración la ecuación (5.4), a la parte izquierda de la ecuación previa se le llamará $\ddot{\mathbf{s}}$ y utilizando la ecuación (5.6), la parte derecha será $\ddot{\mathbf{s}}_d + \mathbf{K}_D\dot{\mathbf{e}}_s + \mathbf{K}_P\mathbf{e}_s$. Por lo tanto:

$$\ddot{\mathbf{e}}_s + \mathbf{K}_D\dot{\mathbf{e}}_s + \mathbf{K}_P\mathbf{e}_s = 0 \tag{5.11}$$

Con la ecuación anterior se consigue un seguimiento asintótico en el espacio imagen. En el Apartado 5.5 se describirán los resultados obtenidos con este controlador durante el seguimiento de distinto tipo de trayectorias.

Debido a la optimización de recursos de la FPGA, llevada a cabo tal y como se describe en el Apartado 5.3, es posible realizar la implementación en la FPGA de distintos controladores visuales. Como experiencia inicial, se implementó un controlador visual basado en Jacobiana transpuesta [Kelly-2000]. Esto permite evaluar las mejoras introducidas haciendo uso del controlador propuesto en tareas de posicionamiento. En este caso el controlador dinámico viene definido por la siguiente ecuación:

$$\boldsymbol{\tau} = \mathbf{L}_J^{\mathrm{T}}\mathbf{K}_P\mathbf{e}_s - \mathbf{K}_D\dot{\mathbf{q}} + \mathbf{g} \tag{5.12}$$

Finalmente, se describe un framework con el que se podrán definir en tiempo de ejecución distintos controladores, una vez se implemente en la FPGA. Este framework se detalla en [Jara-2014], aplicado a tareas de manipulación por parte de manos robóticas, y se describe a continuación para entender qué elementos lo componen y cómo se puede implementar posteriormente en la FPGA.

La función de control que minimiza los pares aplicados a un robot manipulador mientras se ejecuta una tarea de control visual determinada viene definida por:

$$\boldsymbol{\tau} = \mathbf{W}^{-1/2}\left(\mathbf{A}\mathbf{M}^{-1}\mathbf{W}^{-1/2}\right)^{+}\cdot\left(\mathbf{b}-\mathbf{A}\mathbf{M}^{-1}\left(-\mathbf{C}-\mathbf{g}\right)\right) \tag{5.13}$$

donde $\mathbf{W}$ es una matriz de pesos dependiente del tiempo y las m restricciones de la tarea que se debe ejecutar vienen dadas por:

$$\mathbf{A}\left(\mathbf{q},\dot{\mathbf{q}},t\right)\ddot{\mathbf{q}} = \mathbf{b}\left(\mathbf{q},\dot{\mathbf{q}},t\right) \tag{5.14}$$

donde $\mathbf{A} = \mathbf{A}\left(\mathbf{q},\dot{\mathbf{q}},t\right) \in \Re^{mxn}$ y $\mathbf{b} = \mathbf{b}\left(\mathbf{q},\dot{\mathbf{q}},t\right) \in \Re^{mx1}$.

El principal objetivo a alcanzar con los controladores que se quieren derivar de este framework es guiar al robot a lo largo de una trayectoria deseada en el espacio imagen. En este caso, la descripción de la tarea viene determinada por la siguiente ecuación en el espacio imagen:

$$\left(\ddot{\mathbf{s}}_d-\ddot{\mathbf{s}}\right)+\mathbf{K}_D\left(\dot{\mathbf{s}}_d-\dot{\mathbf{s}}\right)+\mathbf{K}_P\left(\mathbf{s}_d-\mathbf{s}\right)=0 \tag{5.15}$$

donde $\mathbf{K}_P$ y $\mathbf{K}_D$ son las ganancias proporcional y derivativa, respectivamente. Esta ecuación se puede expresar con respecto al error en la imagen según la ecuación (5.6). A partir de (5.6) y (5.10) se puede obtener:

$$\ddot{\mathbf{s}}_d+\mathbf{K}_D\dot{\mathbf{e}}_s+\mathbf{K}_P\mathbf{e}_s-\dot{\mathbf{L}}_J\dot{\mathbf{q}}=\mathbf{L}_J\ddot{\mathbf{q}} \tag{5.16}$$

Ahora es posible expresar las restricciones de la tarea de la ecuación (5.16) en la forma de (5.14) con:

$$\begin{aligned}\mathbf{A}&=\mathbf{L}_J\\ \mathbf{b}&=\ddot{\mathbf{s}}_d+\mathbf{K}_D\dot{\mathbf{e}}_s+\mathbf{K}_P\mathbf{e}_s-\dot{\mathbf{L}}_J\dot{\mathbf{q}}\end{aligned} \tag{5.17}$$

Reemplazando las variables concernientes a la descripción de la tarea de imagen, **A** y **b** de (5.17), la ley de control es la siguiente:

$$\boldsymbol{\tau} = \mathbf{W}^{-1/2}\left(\mathbf{L}_J\mathbf{M}^{-1}\mathbf{W}^{-1/2}\right)^{+}\cdot\left(\ddot{\mathbf{s}}_d+\mathbf{K}_D\dot{\mathbf{e}}_s+\mathbf{K}_P\mathbf{e}_s-\dot{\mathbf{L}}_J\dot{\mathbf{q}}-\mathbf{L}_J\mathbf{M}^{-1}\left(-\mathbf{C}-\mathbf{g}\right)\right) \tag{5.18}$$

Tal y como se describirá en el Apartado 5.3 los tres controladores descritos han sido implementados haciendo uso de la arquitectura propuesta basada en FPGA. El interfaz permite al usuario cambiar entre un controlador u otro.

5.3 Implementación en una FPGA de controladores visuales directos

Esta sección describe las principales consideraciones acerca de la implementación basada en FPGA para los controladores visuales directos indicados en las ecuaciones (5.8), (5.12) y (5.18). En el Capítulo 3 se describieron los distintos componentes de la arquitectura semiabierta basada en FPGA propuesta. Como se indicó en el Capítulo 4, los procesamientos visuales llevados a cabo se desarrollan en 5 fases. Resumiendo el proceso, en primer lugar, se recibe la imagen de la tarjeta de captura de imágenes (Fase 1 en la Figura 5.1) para extraer las características visuales empleadas por el sistema de control visual. Los algoritmos de visión artificial son implementados de forma paralela siguiendo el esquema representado en las Fases 2-5 de la Figura 5.1. Así, cada píxel recibido por el módulo de visión se procesa tan pronto como llega al módulo correspondiente, sin la necesidad de almacenar la imagen entera en una memoria. Por lo tanto, no es necesario que se espere a recibir la imagen completa para comenzar con el procesamiento. Así, la latencia del sistema completo se reduce, además de reducir la cantidad de recursos utilizados y las necesidades de comunicación.

La Figura 5.1 muestra el diagrama de tiempos del flujo de datos llevado a cabo por el sistema de control visual en cada ciclo cuando se implementan los controladores (5.2) y (5.8) para el guiado del robot Mitsubishi PA-10. Ya se describió detalladamente el cauce segmentado para el módulo de visión en el Capítulo 4. Ahora se presentan también los tiempos requeridos por el módulo de control. En el caso del controlador implementado para el Mitsubishi PA10, el principal cuello de botella es el tiempo requerido para la transmisión de datos por el protocolo ARCNET (5 ms). Así, el tiempo de captura de la cámara ha sido configurado en consonancia (200 fps). Sin embargo, en el caso de implementar el control para el guiado del robot COOPER, los pares calculados después de la fase de control en el diagrama de tiempos son enviados y aplicados directamente a los amplificadores de los motores del robot. Así, el sistema puede trabajar a la velocidad máxima que permite la cámara utilizada. La arquitectura embebida en FPGA propuesta permitirá obtener un tiempo de ciclo restringido o acotado en el tiempo. Esta es una de las

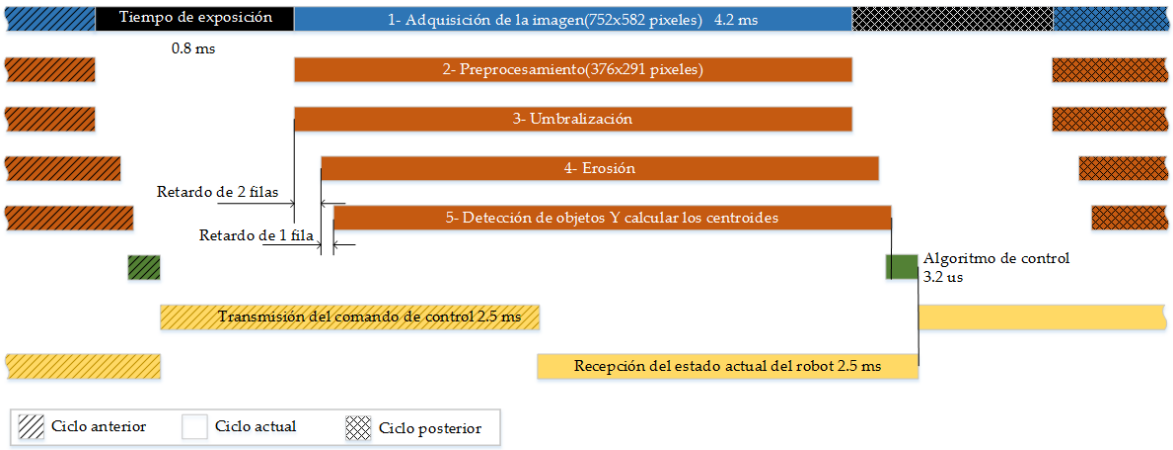


Figura 5.1 Diagrama de tiempos de la ejecución del sistema de control visual en el robot Mitsubishi PA-10

principales ventajas de la arquitectura propuesta frente a los sistemas tradicionales basadas en GPPs. Con la utilización de la arquitectura basada en FPGA el sistema de control visual se comporta como un sistema de tiempo real.

Para implementar el controlador visual propuesto en (5.8) usando la arquitectura basada en FPGA se ha seguido un proceso de optimización haciendo uso del procesamiento paralelo que permite la arquitectura propuesta. La Figura 5.2 muestra el esquema funcional de la implementación del módulo de control implementado en VHDL. Aprovechando las capacidades de paralelismo de la FPGA, se calculan los términos independientes al mismo tiempo, mientras que los dependientes tienen que esperar hasta que las entradas necesarias estén disponibles. Así, tal y como se observa en la Figura 5.2, todos los elementos de las matrices ($\mathbf{L}_J$, $\mathbf{M}$, $\mathbf{C}$, etc.) son calculados simultáneamente, explotando de esta manera las capacidades máximas de ejecución paralela (paralelismo espacial) de la FPGA. Hay que recalcar que, al modificar los grados de libertad del robot utilizado (7 o 3 grados de libertad), el tiempo de cálculo de los algoritmos de control se verán lógicamente afectados. Sin embargo, la ocupación de los recursos hardware de la FPGA se incrementarán exponencialmente. En esta Tesis se ha seguido un método de optimización basado en compartir núcleos (*idle cores*) entre varias operaciones, o ejecutarlos en cauce segmentado.

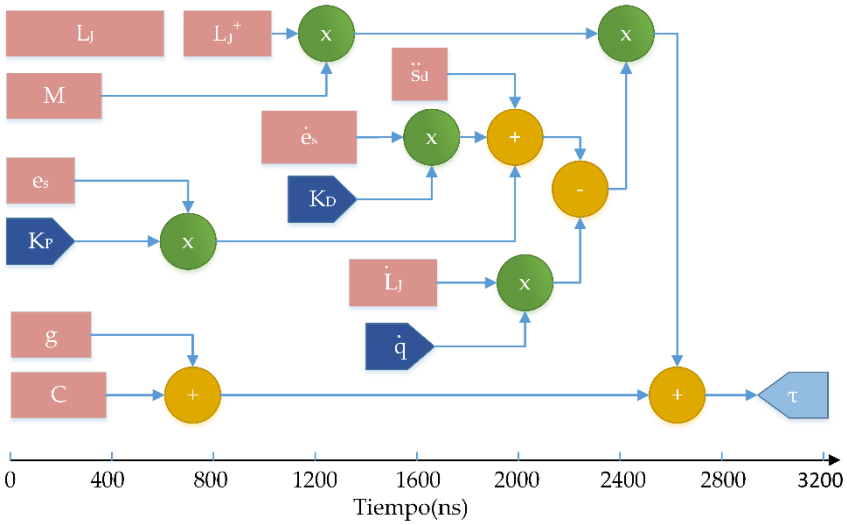


Figura 5.2 Diagrama de tiempos de la ejecución del sistema de control visual propuesto en la Ecuación (5.8)

En la Figura 5.3 se representa el esquema de implementación en VHDL de los distintos módulos que conforman el controlador visual basado en Jacobiana transpuesta.

El diagrama de tiempos para la implementación en VHDL del framework descrito en (5.18) presenta un tiempo de proceso mucho mayor que el empleado por el sistema para calcular los pares articulares de control para (5.8) y (5.12) (ver Figura 5.4). El problema aquí radica en la necesidad de calcular la raíz cuadrada de la matriz **W**. Esta operación consume muchos más ciclos que las operaciones básicas de multiplicación, división o suma/resta. En concreto, para el cálculo de la raíz cuadrada se emplea el algoritmo Blocked Schur [Deadman-2013]. Este método es eficaz y estable y se basa en reducir la

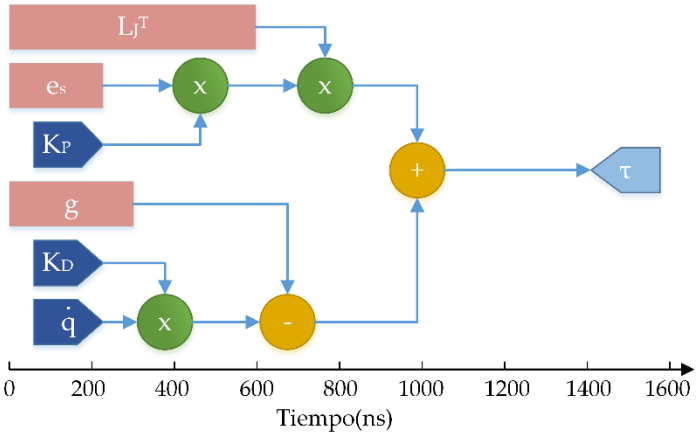


Figura 5.3 Diagrama de tiempos de la ejecución del sistema de control visual basado en Jacobiana transpuesta (Ecuación (5.12))

matriz **W** a la forma triangular de Schur para luego calcular la raíz cuadrada de la matriz triangular. En la Figura 5.4 se muestra únicamente el orden de cálculo, ya que el tiempo invertido en el cálculo de la raíz cuadrada de la inversa de la matriz W es proporcionalmente muy superior al tiempo invertido en el resto de cálculos, con lo que sería imposible mostrar la secuencia de cálculo.

La Tabla 5.1 muestra un resumen de los recursos utilizados por la tarjeta FPGA Kintex-7 de Xilinx. Esta tabla muestra la ocupación de recursos de la FPGA para la implementación de los algoritmos de control para los robots COOPER (3 grados de libertad) y Mitsubishi PA-10 (7 grados de libertad) descritos en (5.8) y (5.12). También se indica el tiempo requerido para la ejecución de dichos algoritmos en ambos robots. El tiempo necesario para la ejecución de los algoritmos de control es prácticamente despreciable comparado con el tiempo requerido en el resto de módulos en la arquitectura propuesta (ver Figura 5.1). Todas las variables empleadas están expresadas en formato punto flotante binario de doble precisión especificado según el estándar IEEE-754. Esta representación emplea 1 bit para signo, 11 bits para exponente y 53 bits para fracción. Para llevar a cabo toda la aritmética de punto flotante en la FPGA se emplea el Xilinx LogiCORE IP Floating-Point Operator v5.0. Este núcleo puede ser configurado dependiendo de la operación, longitud de palabra, latencia e interfaz. Además, este núcleo soporta operaciones como multiplicación, suma/resta, división, comparación, etc., está optimizado en cuanto a velocidad de procesamiento y sigue el estándar IEEE-754. La operación a llevar a cabo se especifica en el momento en el que el núcleo se genera. De esta manera, se genera un núcleo para cada operación. Así, todas las operaciones similares

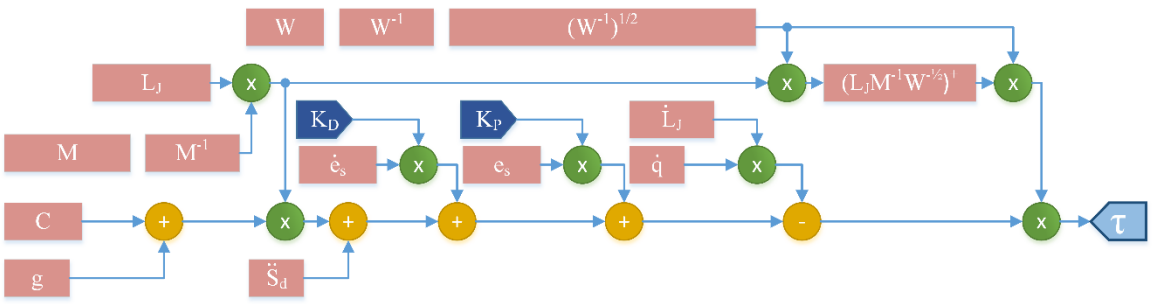


Figura 5.4 Secuencia de cálculo de la ejecución del framework para el diseño en tiempo de ejecución de controladores visuales directos

con la misma matriz comparten el mismo núcleo de operación. Esta técnica es empleada para ahorrar recursos hardware de la FPGA y optimizar la operación. Cada interfaz de operación del núcleo empleado tiene un parámetro denominado "ciclos por operación" que describe el mínimo número de ciclos que debe transcurrir entre entradas. Este ratio puede ser especificado. Un valor de 1 permite que los operandos sean aplicados en cada ciclo de reloj, y, entonces, los resultados se obtienen de manera paralela. Un valor mayor de 1 permite al hardware ser reutilizado. Los recursos consumidos por un núcleo se reducen según incremente el número de ciclos por operación. Un valor de 2 reduce aproximadamente a la mitad el número de recursos. La Tabla 5.2 muestra la latencia de todas las operaciones de punto flotante utilizadas para implementar la ley de control teniendo en cuenta el parámetro "ciclos por operación".

En el caso del control del Mitsubishi PA-10, yna vez que se calculan los pares a aplicar al robot, se mandan al robot a través del módulo de interfaz de ARCNET. Esto consume un total de 5 milisegundos, no sólo para realizar el envío de los pares, sino también para recibir los datos desde el robot para calcular la posición, velocidad y aceleración de cada articulación.

Tabla 5.1 Utilización de Recursos de la FPGA y tiempo de ejecución para los sistemas de control visual propuestos en las Ecuaciones (5.8) y (5.12)

	Slices	LUTs	LUT-RAM	Bloques de DSP	Tiempo de ejecución
Robot 3gdl	6525(12%)	19511(9%)	411(0.4%)	17(1.5%)	2.95 μs
Robot 7gdl	20227(37%)	56581 (26%)	1849(1.8%)	62(5.5%)	3.2 μs

Tabla 5.2 Latencia de las operaciones de punto flotante

Operación	Ciclos por operación	Latencia (ciclo de reloj)
Suma/Resta	1	14
Multiplicación	1	16
División	1	57

5.4 Implementación en una FPGA de controladores visuales directos con compensación de caos

Se dice que se obtiene un comportamiento articular caótico cuando se genera un movimiento impredecible y no periódico en el espacio articular al realizar el seguimiento de una trayectoria repetitiva en la imagen. Este comportamiento articular caótico tiene un gran número de órbitas periódicas inestables (del inglés *Unstable Periodic Orbits*, UPOs). El controlador visual propuesto debe determinar las referencias adecuadas de manera que el efector final del robot y las articulaciones se comporten de manera adecuada (un comportamiento periódico y repetitivo del efector final debe traducirse en un comportamiento también periódico y repetitivo a nivel articular). Para conseguir esto, se integrará un compensador de caos articular en el controlador diseñado en el Apartado 5.2, con el objetivo de eliminar el comportamiento caótico y realizar correctamente el seguimiento de la trayectoria deseada repetitiva en el espacio imagen.

El compensador de caos seleccionado es un controlador de realimentación con retardo DFC. Este método genera una señal de control proporcional a la diferencia entre la velocidad articular actual, $\dot{\mathbf{q}}(t)$ y la velocidad articular retardada por un periodo, $\dot{\mathbf{q}}(t-\varepsilon)$:

$$\boldsymbol{\tau}_{DFC} = \mathbf{M}\lambda\left(\dot{\mathbf{q}}(t-\varepsilon) - \dot{\mathbf{q}}(t)\right) \tag{5.19}$$

donde λ es una constante de ganancia a determinar, ε es el tiempo de retardo de la realimentación (este parámetro es elegido cercano al periodo principal del sistema y corresponde con el UPO seleccionado) y $\mathbf{M}$ es la matriz de inercia del robot. El método de ajuste de estos parámetros ha sido el que se detalla en el artículo [Pomares-2014].

Con estas consideraciones el controlador resultante de añadir a (5.8) el término de compensación de caos es el siguiente:

$$\boldsymbol{\tau} = \boldsymbol{\tau}_{DFC} + \mathbf{M}\mathbf{L}_J^{+} \cdot \left(\ddot{\mathbf{s}}_d + \mathbf{K}_D\dot{\mathbf{e}}_s + \mathbf{K}_P\mathbf{e}_s - \dot{\mathbf{L}}_J\dot{\mathbf{q}}\right) + \mathbf{C} + \mathbf{g} \tag{5.20}$$

La Figura 5.5 muestra el esquema funcional de la implementación del módulo de control con compensación de caos en VHDL. Dado que el controlador de Jacobiana transpuesta descrito en [Kelly-2000] e implementado a través de su ley de control en (5.12) sobre la FPGA (como se veía en la Figura 5.3), no permite realizar seguimientos de

trayectorias, al ser un controlador básico de posicionamiento o regulación, no tiene sentido añadir el término de compensación de caos. Sí que se ha implementado el cálculo de este término en el framework que permite obtener toda una serie de controladores visuales directos simplemente definiendo nuevos valores de la matriz **W**. La Figura 5.6 muestra el esquema de implementación VHDL con las secuencias seguidas para el cálculo de los distintos términos del controlador descrito por la ley de control mostrada en la Ecuación (5.18).

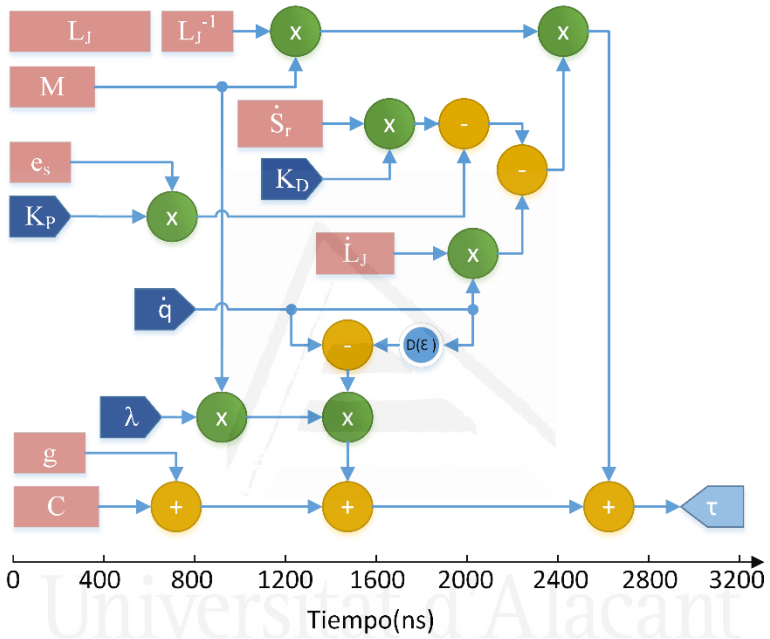


Figura 5.5 Diagrama de tiempos de la ejecución del sistema de control visual propuesto con compensación de caos

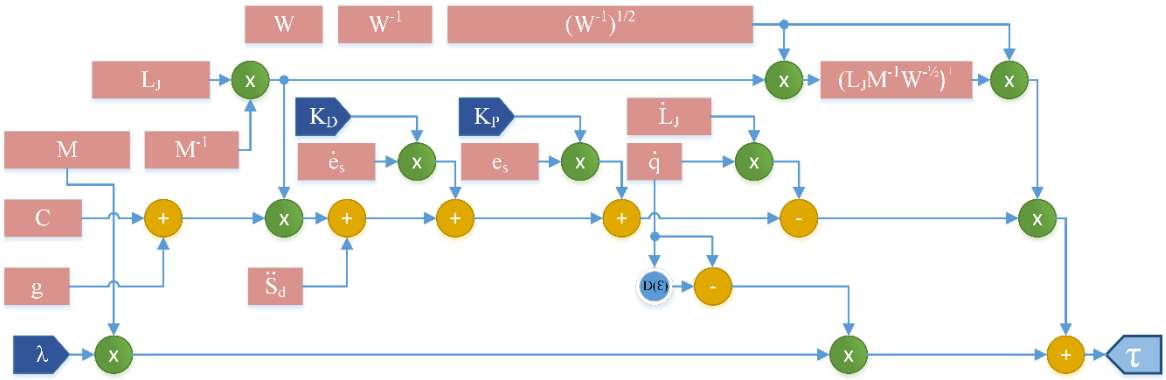


Figura 5.6 Secuencia de la ejecución del framework para el diseño en tiempo de ejecución de controladores visuales directos con compensación de caos

5.5 Resultados

Esta sección presenta diferentes experimentos con el objetivo de ilustrar el comportamiento del sistema de control visual propuesto, así como su implementación en la arquitectura diseñada basada en FPGA. La arquitectura y el sistema de control visual directo propuesto son evaluados en dos robots: el robot COOPER de 3 grados de libertad y el robot Mitsubishi PA-10 de 7 grados de libertad. En primer lugar, se presenta el comportamiento del sistema con el robot COOPER. A continuación, se presentan los resultados obtenidos durante el seguimiento de trayectorias más complejas empleando el robot Mitsubishi PA-10.

Los parámetros intrínsecos considerados para la cámara son (u_0, v_0) = (368, 290) px, y (f_u, f_v) = (433, 433) px (posición del centro óptico (u_0, v_0) y la longitud focal en las direcciones X e Y respectivamente).

5.5.1 Control visual del robot COOPER de 3 grados de libertad

En este apartado se describe una serie de experimentos realizados con el robot de 3 gdl COOPER. COOPER es un robot planar como demuestra la con ejes paralelos y, situando la cámara en su extremo mirando hacia un plano 2D paralelo, sus articulaciones permiten el posicionamiento del extremo (o de la cámara, ya que la relación de los sistemas de referencia de ésta con respecto al efector final será fija y conocida) en ese plano paralelo al robot. Así, los experimentos de control se centran en posicionamientos y seguimiento de trayectorias en el plano XY. Habitualmente se indica que son necesarios al menos 3 puntos característicos en la imagen para desarrollar una tarea de control visual, pero en este caso sería necesario un robot de al menos 6 grados de libertad. Sin embargo, para posicionamientos en el plano XY, sólo se requieren 2 grados de libertad. La tarea de control visual en este caso puede resolverse con un único punto característico (k = 1). Por ello, en esta sección se utiliza una característica visual para los experimentos de guiado del robot. Se emplea un patrón con un objeto blanco sobre un fondo negro para simplificar los procesamientos en la imagen en estos experimentos iniciales.

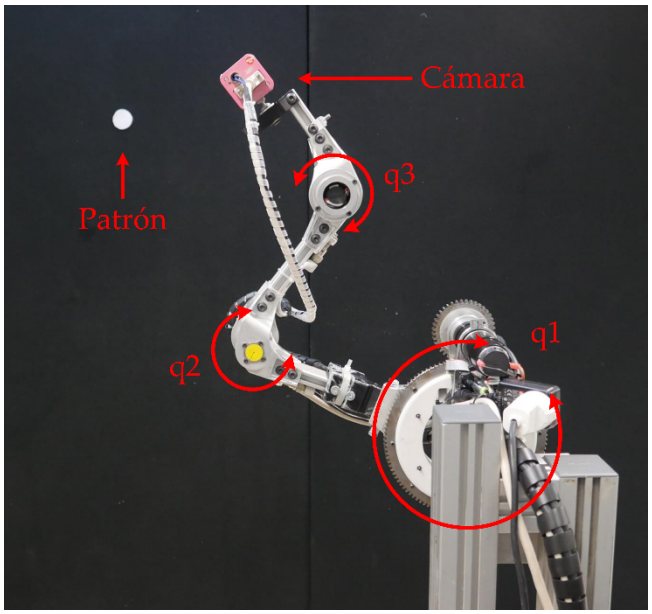


Figura 5.7 Montaje experimental para la evaluación de la arquitectura en el robot COOPER.

5.5.1.1 Experimento 1

Este experimento constituye la primera implementación desarrollada de un controlador visual directo basado en FPGA. En concreto, se ha implementado un controlador visual directo clásico, como es el basado en Jacobiana transpuesta. Para llevar a cabo el experimento, en primer lugar, el robot se posiciona en la localización final deseada. En esta localización, se realiza el procesamiento de la imagen obteniendo el centroide del objeto observado desde la localización final deseada. Este centroide constituirá el valor final de la característica deseada empleada como referencia por el sistema de control visual. A continuación el robot se mueve a la posición inicial, obteniendo el valor de la característica en la imagen desde la posición inicial. Entre ambas posiciones (inicial y deseada) se define la trayectoria a seguir.

En este experimento, para las ganancias que se representan en la Ecuación (5.12) se emplean los siguientes valores:

$$\mathbf{K}_P = \begin{pmatrix} 0.01 & 0 \\ 0 & 0.07 \end{pmatrix};\ \mathbf{K}_D = \begin{pmatrix} 0.3 & 0 & 0 \\ 0 & 0.2 & 0 \\ 0 & 0 & 0.3 \end{pmatrix} \tag{5.21}$$

Haciendo uso de las ganancias anteriores se obtienen los resultados que se muestran en la Figura 5.8 para una tarea de posicionamiento usando el controlador basado en

Jacobiana transpuesta. Esta figura muestra la evolución del centro de gravedad del objeto en la imagen así como la posición 3D del extremo de la cámara ubicada en el extremo del robot. Se observa que, si bien se consigue realizar el posicionamiento de forma correcta, no se consiguen eliminar por completo las oscilaciones en el tramo final de la trayectoria. Con el objetivo de representar más claramente el comportamiento del sistema, en la Figura 5.9 se indica la evolución del módulo del error en la imagen, así como los pares articulares aplicados.

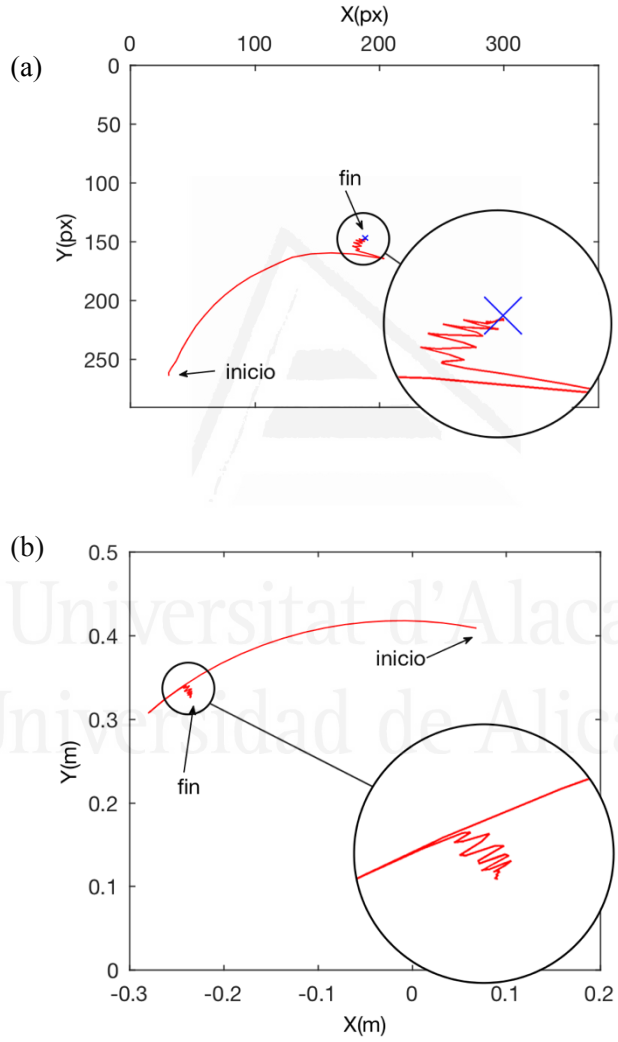


Figura 5.8 Experimento 1: Robot COOPER. a) Evolución del centro de gravedad del objeto en el plano imagen. b) Evolución de la posición del efector final del robot en el espacio Cartesiano 3D

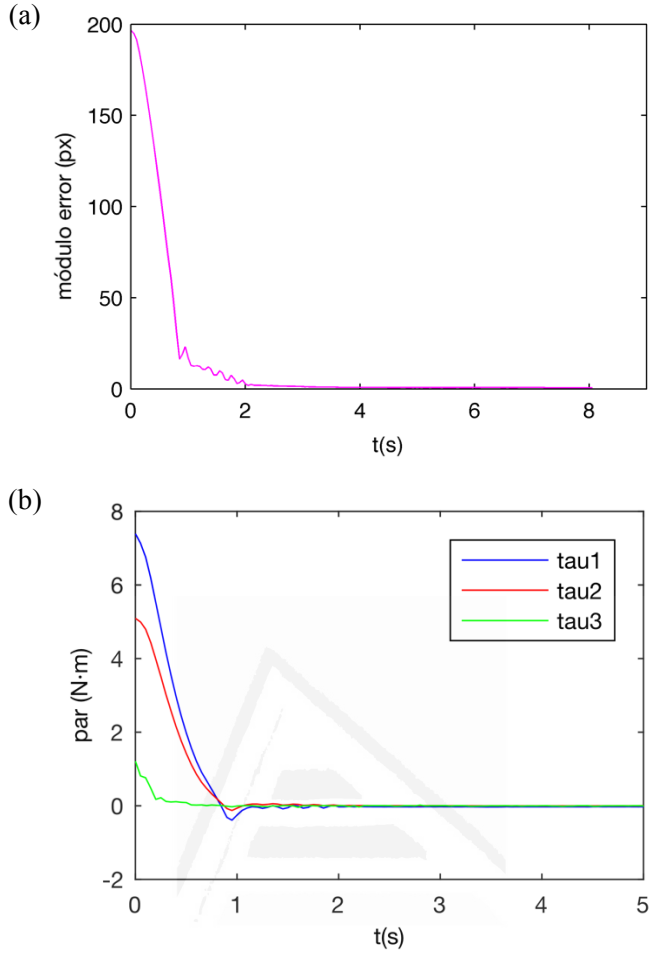


Figura 5.9 Experimento 1: Robot COOPER. a) Evolución del módulo del error en la imagen durante el seguimiento. b) Evolución de la salida del sistema: pares articulares.

5.5.1.2 Experimento 2

Las condiciones de ejecución de este experimento son similares al representado en el Experimento 1, sin embargo, en este caso se empleará el controlador directo propuesto en la Ecuación (5.8). Al igual que en el caso anterior, se considera la implementación propuesta basada en FPGA. Con el objetivo de obtener velocidades de ejecución similares se consideran, para las ganancias que se representan en la Ecuación (5.8), los siguientes valores:

$$\mathbf{K}_P = \begin{pmatrix} 0.7 & 0 \\ 0 & 1.2 \end{pmatrix};\ \mathbf{K}_D = \begin{pmatrix} 0.25 & 0 \\ 0 & 0.3 \end{pmatrix} \tag{5.22}$$

Dado que lo que se quiere en este experimento es comparar el comportamiento del controlador descrito en la Ecuación (5.8) con el método de Jacobiana transpuesta, y este

controlador es incapaz de realizar un seguimiento de una trayectoria (se trata de un controlador de posicionamiento puro o regulador), se considera en este experimento un vector de características deseadas, $\mathbf{s}_d$, constante. La Figura 5.10.a muestra la evolución del centro de gravedad del objeto en el plano imagen durante el experimento. Frente a los resultados mostrados en el ejemplo anterior, se observa que se parte de la posición inicial y se alcanza la posición final deseada sin oscilaciones. La Figura 5.10.b muestra la trayectoria seguida por el efector final del robot durante el seguimiento llevado a cabo por el controlador propuesto. El robot mueve la cámara siguiendo la trayectoria deseada (se aproxima a una trayectoria parabólica) sin oscilaciones. Una de las figuras que mejor representa la correcta ejecución del sistema de control visual es la Figura 5.11.a. Esta

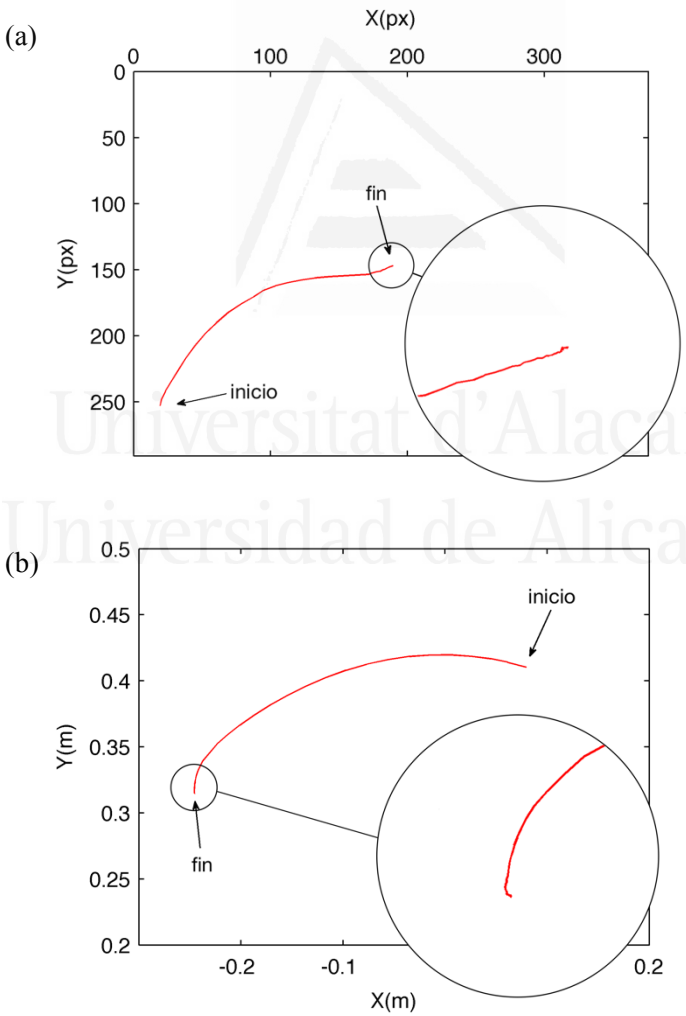


Figura 5.10 Experimento 2: Robot COOPER. a) Evolución del centro de gravedad del objeto en el plano imagen. b) Evolución de la posición del efector final del robot en el espacio Cartesiano 3D

última figura muestra la evolución del módulo del error en la imagen durante el seguimiento. Este error es calculado haciendo uso del centro de gravedad del objeto observado (diferencia entre la posición actual en la imagen y la deseada). El error decrece rápidamente en las primeras iteraciones. Finalmente, la Figura 5.11.b muestra las acciones de control generadas por el controlador durante la tarea de seguimiento, es decir, los pares enviados a los motores de cada una de las articulaciones del robot. Esta tarea se lleva a cabo durante 2 segundos aproximadamente y, una vez alcanzada la posición final, el controlador genera pares casi nulos.

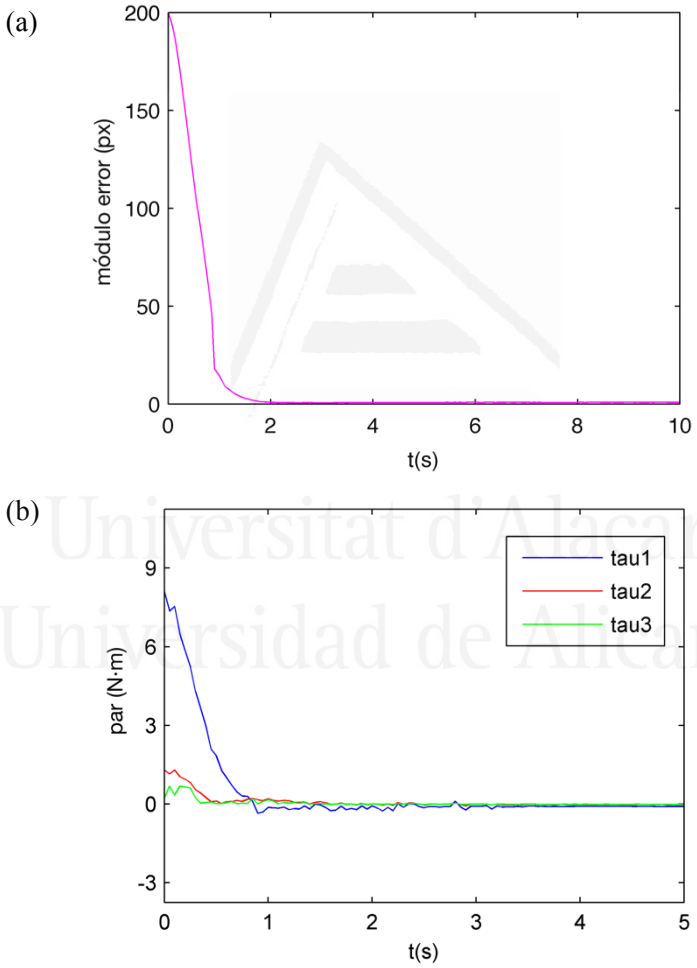


Figura 5.11 Experimento 2: Robot COOPER. a) Evolución del módulo del error en la imagen durante el seguimiento. b) Evolución de la salida del sistema: pares articulares

5.5.1.3 Experimento 3

Este experimento evalúa la implementación del controlador propuesto en (5.8) sobre la FPGA durante el seguimiento de una trayectoria en imagen repetitiva. Dicha trayectoria deseada para la característica extraída en la imagen viene definida por la siguiente ecuación:

$$\mathbf{s}_d = \begin{bmatrix} f_{xd} \\ f_{yd} \end{bmatrix} = \begin{bmatrix} 320 + 166\cos(\omega t + \pi/4) + 20\sin(\omega 5t) \\ 265 + 160\sin(\omega t + \pi/4) + 15\sin(\omega 5t) \end{bmatrix} \tag{5.23}$$

En la Figura 5.12 se representa la trayectoria deseada en el espacio de la imagen definida haciendo uso de la Ecuación (5.23). En la Figura 5.13 se observa el comportamiento obtenido durante el seguimiento considerando una velocidad media/alta (ω = 1 rad/s). En este experimento se han considerado los siguientes valores para las ganancias del controlador $\mathbf{K}_P = 0.1\mathbf{I}_{2x2}$ y $\mathbf{K}_D = 0.5\mathbf{I}_{2x2}$ (ganancias proporcional y derivativa para la trayectoria deseada a seguir). La Figura 5.13 representa en rojo la trayectoria deseada y en azul la obtenida durante el seguimiento. Se observa que se obtiene un correcto comportamiento y los errores de seguimiento son mínimos. Por último, en la Figura 5.14 se representan los pares articulares aplicados por el robot y que permiten llevar a cabo correctamente el seguimiento.

Con el objetivo de evaluar el comportamiento del controlador implementado cuando se aplican distintas velocidades de seguimiento, la Tabla 5.3 representa el error medio en píxeles cuando el parámetro ω de la Ecuación (5.23) vale 0.5, 1 y 2 rad/s, respectivamente.

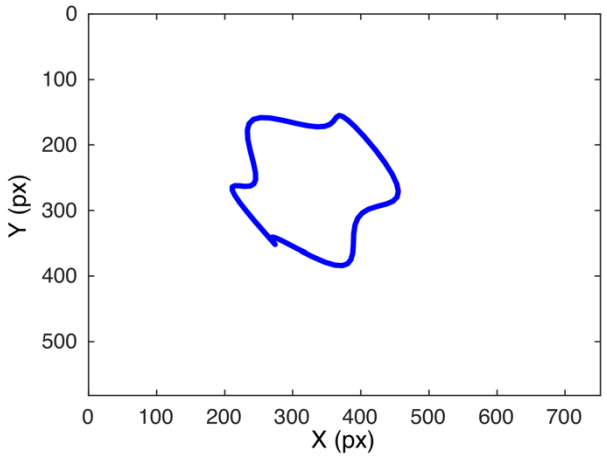


Figura 5.12 Experimento 3: Robot COOPER. Trayectoria deseada en la imagen

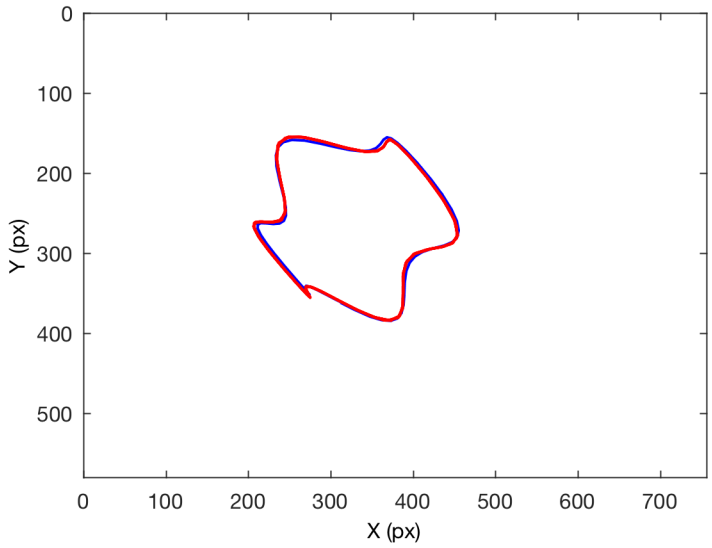


Figura 5.13 Experimento 3: Robot COOPER. Trayectoria obtenida en la imagen durante el experimento.

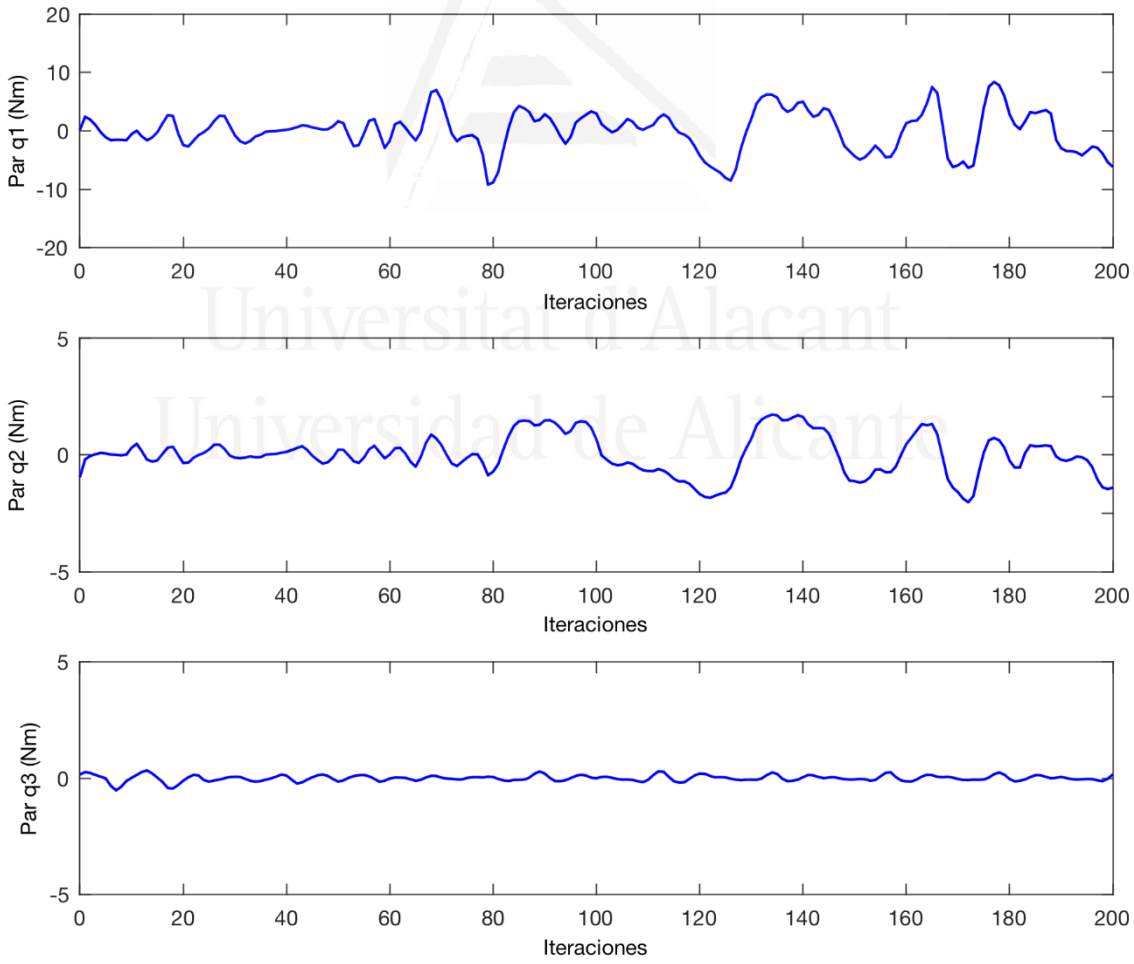


Figura 5.14 Experimento 3: Robot COOPER. Pares articulares durante el experimento

Tabla 5.3 Experimento 3: Robot COOPER. Error medio en píxeles considerando distintas velocidades de seguimiento

ω = 0.5 rad/s	ω = 1 rad/s	ω = 2 rad/s
2.14	2.55	3.1

Un aspecto importante a tener en cuenta en la implementación de cualquier software embebido en una FPGA es la ocupación de los recursos hardware. La Tabla 5.4 muestra un resumen de los recursos utilizados de la tarjeta Kintex-7 de Xilinx empleada en esta Tesis. Este resumen muestra la ocupación de recursos necesarios para la implementación de los controladores descritos en la Ecuación (5.8) y (5.12). La optimización realizada ha permitido la implementación del sistema de control visual directo haciendo uso tan sólo del 21% de las puertas lógicas disponibles en la Kintex-7. Este es uno de los recursos más utilizados en la implementación. El segundo recurso más utilizado han sido las LUTs. El resto se encuentra por debajo del 5% de uso. El esfuerzo de optimización llevado a cabo permite crear nuevos controladores en la misma arquitectura implementando todos al mismo tiempo. De esta manera, el usuario podría cambiar entre uno u otro dependiendo de, por ejemplo, la trayectoria a seguir, el robot, la configuración del experimento, etc.

La Figura 5.15 muestra el diagrama de tiempos del sistema donde se puede observar el flujo de ejecución de los distintos módulos que integran la implementación del sistema de control visual directo basado en imagen propuesto. Tal y como se puede apreciar en esta figura, el flujo de datos de los distintos módulos se ejecuta en paralelo, lo cual constituye una de las principales ventajas de implementar el controlador en la FPGA. Ya en el Capítulo 4 se mostró este cauce segmentado para el módulo de visión. Aquí se presenta también el módulo de control. La Figura 5.15 indica la latencia requerida en cada fase. Los módulos de visión están sincronizados con la señal de reloj de pixel de la cámara PCLK. La cámara manda los datos de imagen con una frecuencia de 80 MHz, enviando dos pixels al mismo tiempo cada ciclo de reloj. La cámara ha sido configurada para capturar las imágenes a la máxima resolución (752x582 píxeles) y con un ratio de captura de 340 fps. De esta manera, el ratio de adquisición de la imagen completa es de 2.9 ms. Una vez

Tabla 5.4 Experimento 3: Robot COOPER. Ocupación de recursos de la FPGA

	Slices	LUTs	LUT-RAM	Block RAM/FIFO	DSP Blocks
Algoritmos de visión	327	7865	400	49	6
Algoritmo de control	6525	19511	411	0	17
Otros	1572	4572	15	0	0
Total	10834	31948	826	49	23
Ocupación FPGA	21%	15%	1%	4%	2%

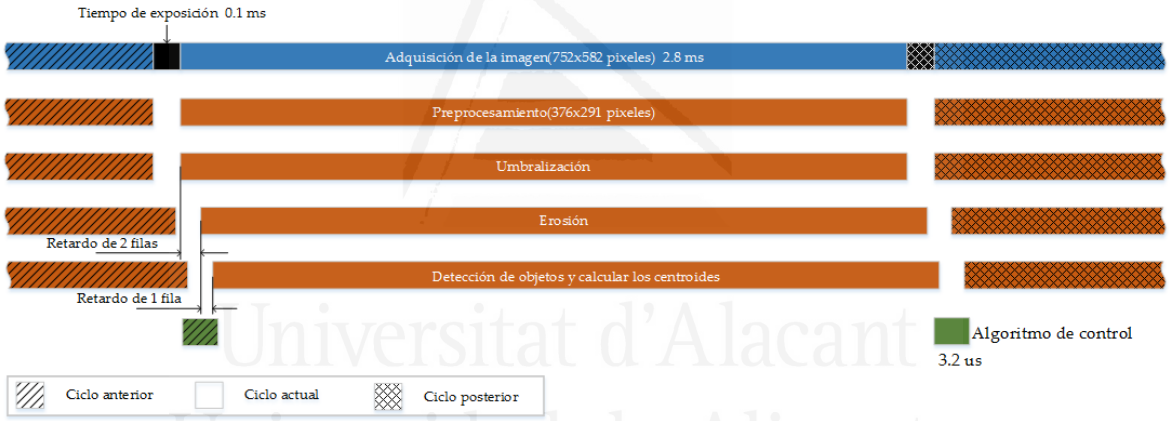


Figura 5.15 Experimento 3: Diagrama de tiempos de la ejecución del sistema de control visual en el robot COOPER

que se han detectado los centroides de los objetos, se mandan al módulo de control para tomar las decisiones o generar los pares articulares adecuados para mover el robot de manera que se siga el objeto según la trayectoria deseada. En la Figura 5.2 se mostró el diagrama de tiempos requeridos para el cálculo de los distintos componentes que integran el módulo de control (5.8). El tiempo requerido para generar los pares (3.2 us) es despreciable comparado con el tiempo necesario para realizar el proceso de captura y procesamiento de las imágenes. Para (5.2) el tiempo de cálculo necesario es incluso menor, de 1.6 us.

5.5.1.4 Experimento 4

Con el objetivo de evaluar las mejoras obtenidas al emplear la arquitectura de control basada en FPGA, esta sección compara el comportamiento cuando se aplica el controlador con compensación de caos descrito en el Apartado 5.4 usando la arquitectura propuesta frente a implementaciones clásicas como la descrita en [Pomares-2014]. En concreto, se va a evaluar durante el seguimiento de la trayectoria repetitiva en la imagen descrita en la Figura 5.12.

La Figura 5.16 muestra el seguimiento en el espacio imagen cuando se aplica la ley de control propuesta (con compensación de caos), utilizando la arquitectura propuesta basada en FPGA, y sin utilizarla. Se observa que en ambos casos se obtiene un comportamiento correcto durante el seguimiento de la trayectoria en el espacio imagen (ω = 1 rad/s). Usando el método de ajuste de parámetros del controlador DFC descrito en [Pomares-2014], se obtiene un valor de λ de 4.6. Aunque en ambos casos el resultado es adecuado, usando una implementación clásica, el módulo del error medio en el espacio imagen es de 3.82 píxeles, mientras que el valor de este parámetro para la implementación basada en FPGA es de 2.55 píxeles. El valor de esta reducción en el error es debido, no sólo a la reducción de los tiempos de procesamiento, sino también a la posibilidad de obtener tiempos de ciclos constantes. Esto último permite realizar un mejor ajuste y optimización del controlador DFC. Para observar más claramente las mejoras, a

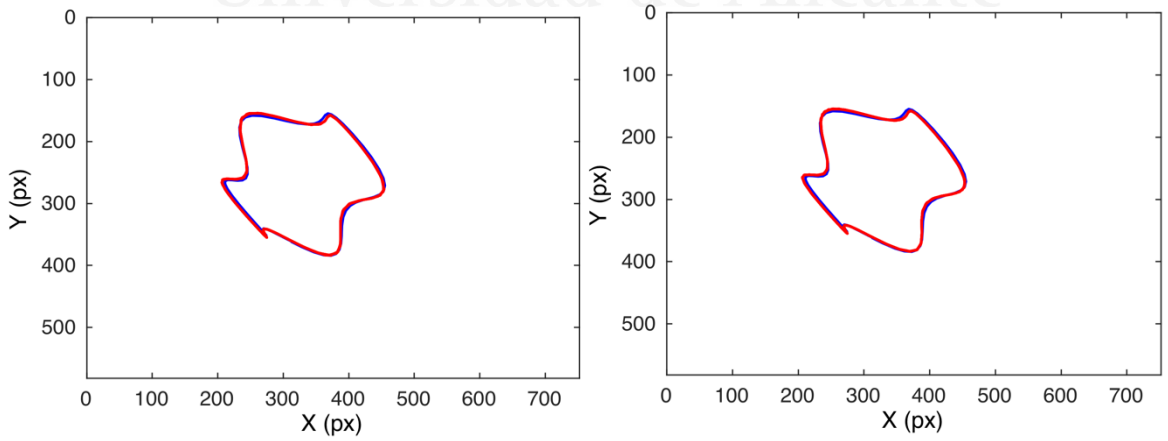


Figura 5.16 Experimento 4: Robot COOPER. Comportamiento en el espacio imagen. **(a)** sin utilizar la arquitectura basada en FPGA. **(b)** Utilizando la arquitectura propuesta

continuación, se analiza el comportamiento en el espacio articular. La Figura 5.17.a representa el comportamiento articular usando una implementación clásica (no FPGA), mientras que la Figura 5.17.b muestra las mismas figuras cuando se hace uso de la arquitectura propuesta. El objetivo del controlador es obtener un comportamiento periódico articular durante el seguimiento de una trayectoria periódica en el espacio imagen. Usando una arquitectura clásica se obtiene un comportamiento adecuado. Sin embargo, debido a las variaciones en los tiempos de ciclo debido a los retardos en la realimentación, el ajuste no es del todo correcto. Como se observa en la figura, haciendo uso de la arquitectura basada en FPGA, es posible obtener y predecir de antemano con precisión los retardos en la realimentación. Esto permite un mejor ajuste del controlador y obtener un comportamiento más adecuado en el espacio articular.

5.5.1.5 Experimento 5

Como se observa en la Ecuación (5.18), el framework para el diseño de controladores visuales directos depende de la matriz **W**. Esta matriz determina cómo se distribuyen los pares articulares en las articulaciones. Considerando diferentes valores para esta matriz, se pueden obtener controladores visuales directos basados en imagen conocidos, al tiempo que se pueden obtener nuevos controladores con comportamientos dinámicos distintos. Por ejemplo, si se añade el término de compensación de caos y se elige $\mathbf{W} = \mathbf{M}^{-2}$, se obtiene el controlador propuesto en (5.8) e implementado ya en la FPGA, tal y como se describe en los apartados anteriores.

Otros valores interesantes para la matriz **W** son **I** y **D**, siendo **I** la matriz identidad y **D** una matriz diagonal positiva. Al elegir la matriz **I** se simplifica la ley de control:

$$\boldsymbol{\tau} = \left(\mathbf{L}_J\mathbf{M}^{-1}\right)^{+} \cdot \left(\ddot{\mathbf{s}}_d + \mathbf{K}_D\dot{\mathbf{e}}_s + \mathbf{K}_P\mathbf{e}_s - \dot{\mathbf{L}}_J\dot{\mathbf{q}} - \mathbf{L}_J\mathbf{M}^{-1}\left(-\mathbf{C}-\mathbf{g}\right)\right) + \boldsymbol{\tau}_{DFC} \tag{5.24}$$

Eligiendo $\mathbf{W} = \mathbf{D}$, se permite distribuir los pares en las articulaciones, de forma que pesos grandes causarán pares pequeños en las articulaciones correspondientes:

$$\boldsymbol{\tau} = \mathbf{D}^{-1/2}\left(\mathbf{L}_J\mathbf{M}^{-1}\mathbf{D}^{-1/2}\right)^{+} \cdot \left(\ddot{\mathbf{s}}_d + \mathbf{K}_D\dot{\mathbf{e}}_s + \mathbf{K}_P\mathbf{e}_s - \dot{\mathbf{L}}_J\dot{\mathbf{q}} - \mathbf{L}_J\mathbf{M}^{-1}\left(-\mathbf{C}-\mathbf{g}\right)\right) + \boldsymbol{\tau}_{DFC} \tag{5.25}$$

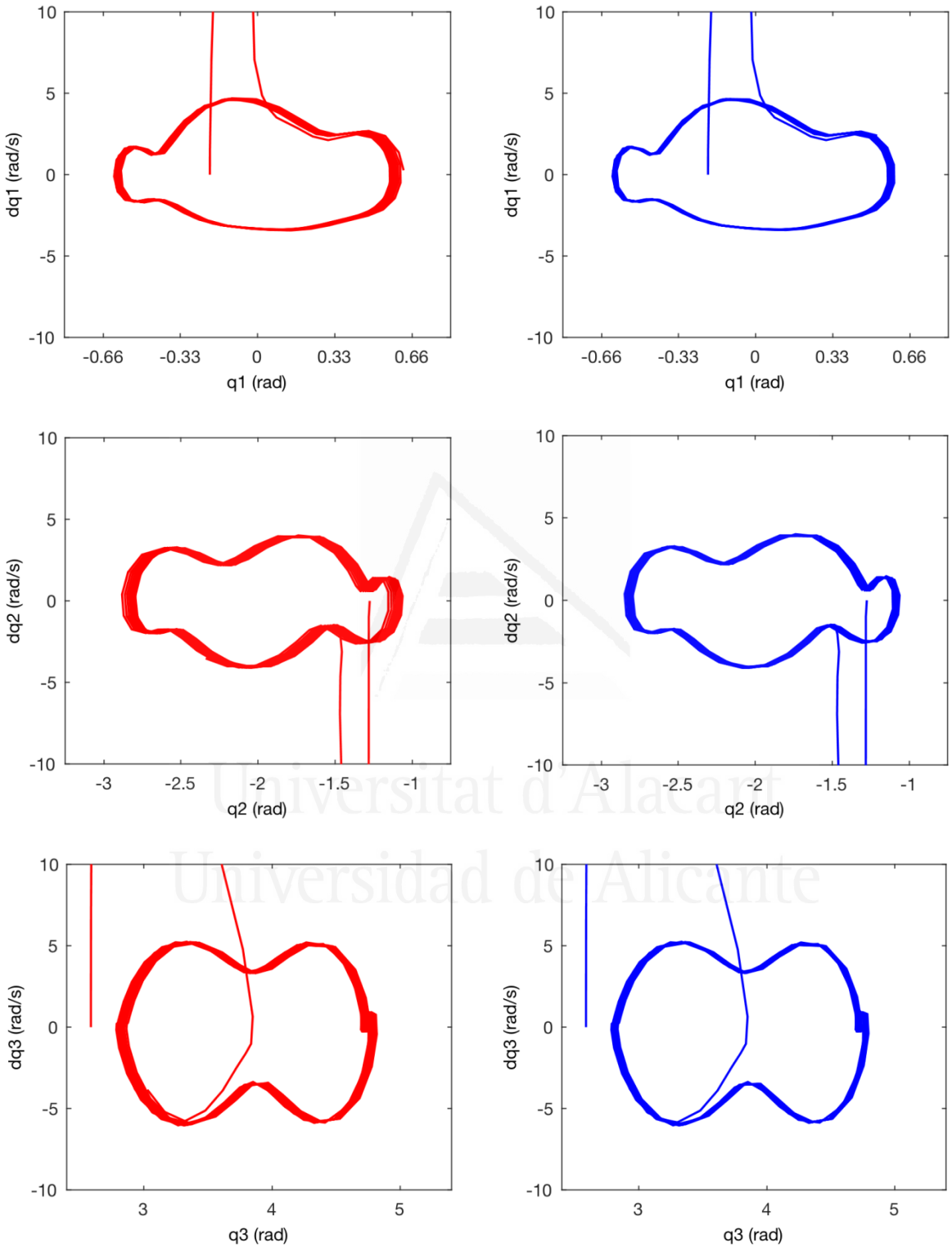


Figura 5.17 Experimento 4: Robot COOPER. Comportamiento en el espacio articular: sin usar la arquitectura basada en FPGA (izquierda), y utilizando la arquitectura basada en FPGA (derecha)

Como se describe en [Udwadia-2003], otro valor que resulta interesante considerar para **W** es $\mathbf{M}^{-1}$, ya que es consistente con el principio d'Alembert. La ley de control obtenida en este caso es igual a:

$$\boldsymbol{\tau} = \mathbf{M}^{1/2}\left(\mathbf{L}_J\mathbf{M}^{-1}\mathbf{M}^{1/2}\right)^{+}\cdot\left(\ddot{\mathbf{s}}_d + \mathbf{K}_D\dot{\mathbf{e}}_s + \mathbf{K}_P\mathbf{e}_s - \dot{\mathbf{L}}_J\dot{\mathbf{q}} - \mathbf{L}_J\mathbf{M}^{-1}\left(-\mathbf{C}-\mathbf{g}\right)\right) + \boldsymbol{\tau}_{DFC} \tag{5.26}$$

Considerando la definición de la pseudoinversa de una matriz se puede obtener:

$$\boldsymbol{\tau} = \mathbf{L}_J^T\left(\mathbf{L}_J\mathbf{M}^{-1}\mathbf{L}_J^T\right)^{1}\cdot\left(\ddot{\mathbf{s}}_d + \mathbf{K}_D\dot{\mathbf{e}}_s + \mathbf{K}_P\mathbf{e}_s - \dot{\mathbf{L}}_J\dot{\mathbf{q}} - \mathbf{L}_J\mathbf{M}^{-1}\left(-\mathbf{C}-\mathbf{g}\right)\right) + \boldsymbol{\tau}_{DFC} \tag{5.27}$$

Gracias a la arquitectura abierta embebida en FPGA descrita en el Capítulo 3, ha sido posible implementar este framework representado por la ecuación (5.18). Considerando los valores indicados previamente para **W**, se ha realizado el mismo experimento de seguimiento de trayectoria en el espacio imagen descrito por la Ecuación (5.23). Cabe mencionar que otros valores de **W** pueden proporcionar al controlador obtenido diferentes propiedades dinámicas durante el seguimiento de la trayectoria en la imagen. Tal como se describió en el Capítulo 3, se ha desarrollado también un interfaz que permite monitorizar la tarea de control visual al mismo tiempo que proporciona la posibilidad de modificar diversos parámetros y constantes de los controladores implementados en el hardware de la FPGA. Uno de estos parámetros disponibles para el usuario en tiempo de ejecución es el ajuste de la matriz **W**, sin la necesidad de recompilar el programa en la FPGA.

Como muestra la Figura 5.18, se obtiene un correcto seguimiento de la trayectoria deseada en todos los casos cuando se emplea una velocidad alta (ω = 1 rad/s). El mejor comportamiento lo proporciona **W** = $\mathbf{M}^{-1}$, aunque el seguimiento de la trayectoria es similar al obtenido con el controlador propuesto en (5.8), tal y como puede apreciarse en el Apartado 5.5.1.4. En todos los experimentos mostrados en este apartado, la ganancia proporcional ha sido establecida en $\mathbf{K}_P = 0.1\mathbf{I}$, y la ganancia derivativa en $\mathbf{K}_D = 0.5\mathbf{I}$. Dado que es complicado ver en las gráficas de la Figura 5.18 el comportamiento de uno y otro controlador, en la Figura 5.19 se muestran las acciones de control o pares generados

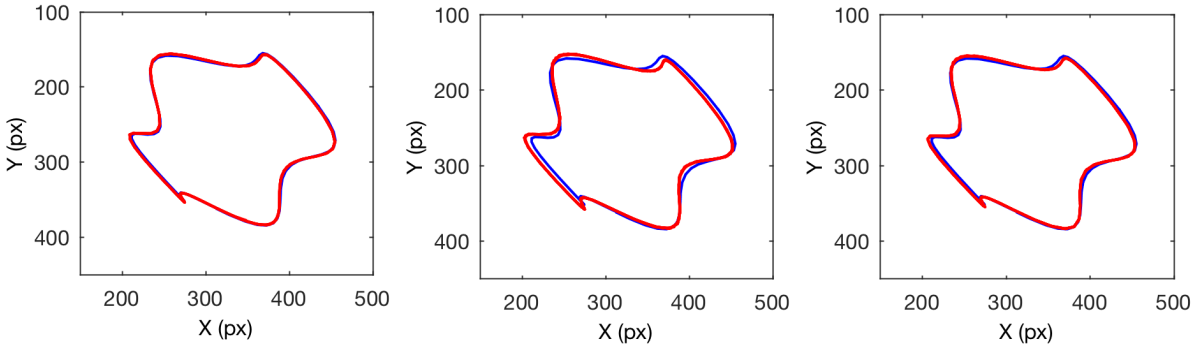


Figura 5.18 Experimento 5: Trayectoria deseada en la imagen (azul) y trayectoria obtenida (rojo): **(a)** $\mathbf{W} = \mathbf{M}^{-1}$; **(b)** $\mathbf{W} = \mathbf{D}$; **(c)** $\mathbf{W} = \mathbf{I}$

durante el experimento. La matriz **D** es una matriz diagonal donde el peso correspondiente a la tercera articulación se ha establecido al doble del peso correspondiente a la primera y la segunda articulaciones. Se observa en la Figura 5.19 el efecto que se obtiene con esta matriz. Si comparamos la Figura 5.19.b con la Figura 5.19.c se puede observar que cuando $\mathbf{W} = \mathbf{D}$ se obtienen pares más pequeños en la tercera articulación (tal como se esperaba, ya que el peso correspondiente a esta articulación era dos veces mayor que los pesos de las otras dos articulaciones). Así, esta matriz diagonal se puede emplear para distribuir los pares y disminuir el esfuerzo en las articulaciones que se desee.

La Tabla 5.5 muestra el error medio en píxeles durante el seguimiento, considerando los controladores previos generados a partir del framework, y con diferentes velocidades de seguimiento. Como se ha indicado anteriormente, el mejor comportamiento se obtiene cuando $\mathbf{W} = \mathbf{M}^{-1}$. Si comparamos con la Tabla 5.3, podemos ver que se mejora el comportamiento de este último controlador con el controlador propuesto en (5.8). Como contraste, cuando $\mathbf{W} = \mathbf{I}$, el controlador es mucho más simple y fácil de implementar en la FPGA. El uso de $\mathbf{W} = \mathbf{D}$ permite indicar qué articulaciones soportarán las mayores cargas. Así, el framework implementado en la FPGA permite en tiempo de ejecución ajustar diversos comportamientos del seguimiento de trayectorias para adecuarlo a cada situación. Además, el ajuste mediante la interfaz de **W** permite probar nuevos controladores fácilmente para buscar siempre el más adecuado a las propiedades de la trayectoria deseada en imagen.

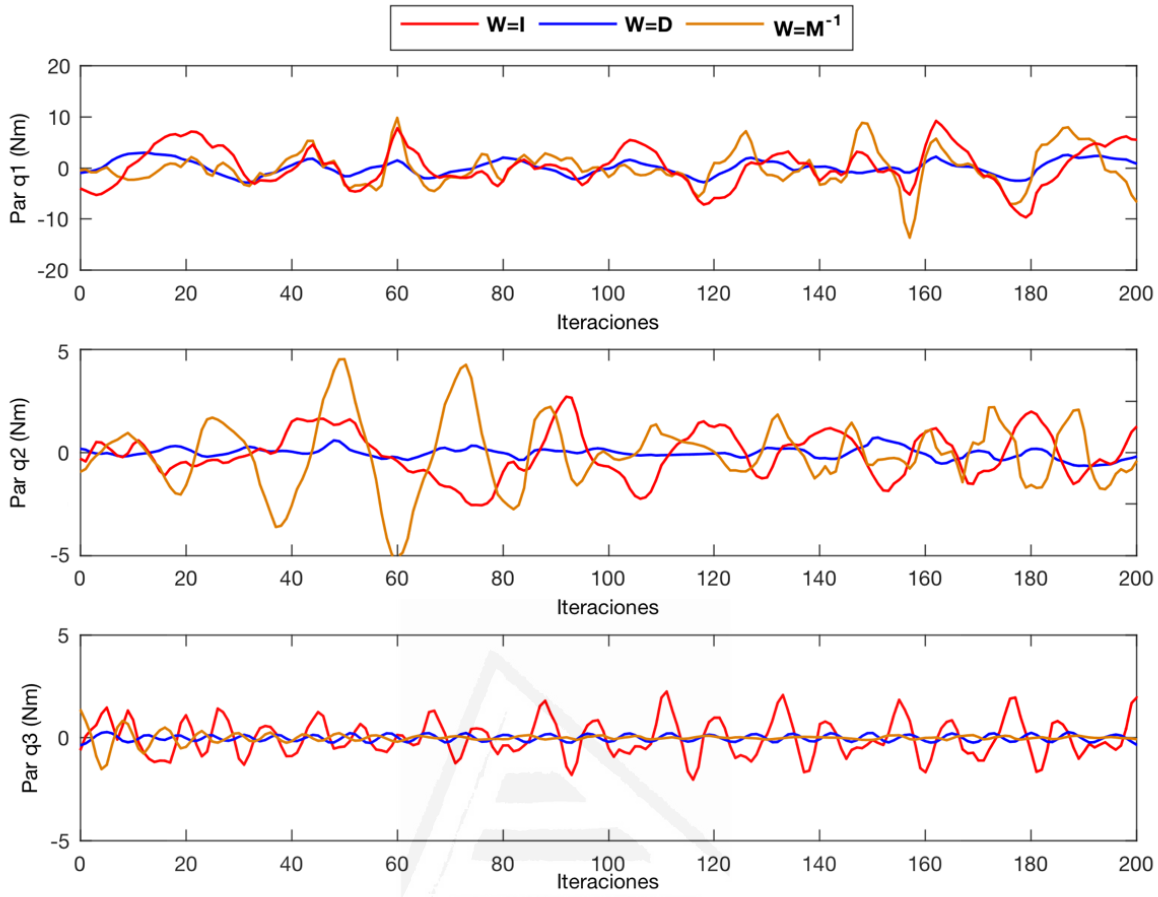


Figura 5.19 Experimento 5: Pares articulares durante el experimento: **W** = $\mathbf{M}^{-1}$ (azul); **W** = **D** (naranja); **W** = **I** (rojo).

Tabla 5.5 Error medio en píxeles en el seguimiento de la trayectoria considerando distintas velocidades

W	ω = 0.5 rad/s	ω = 1 rad/s	ω = 2 rad/s
$\mathbf{M}^{-1}$	2.23	2.8	3.52
I	2.6	4.14	5.1
D	3.95	5.73	6.1

La implementación de este framework se ha mostrado de gran interés para el diseñador de controladores visuales directos. Sin embargo, el precio a pagar en este caso es el aumento considerable de los recursos utilizados en la FPGA. El esfuerzo de optimización ha permitido integrar completamente el framework en la FPGA, al tiempo que se dejan recursos libres para el desarrollo de nuevos controladores o funcionalidades en la arquitectura semiabierta propuesta. La Tabla 5.6 resume la ocupación de los recursos

Tabla 5.6 Experimento 5: Robot COOPER. Ocupación de recursos de la FPGA

	Slices	LUTs	LUT-RAM	Block RAM/FIFO	DSP Blocks
Algoritmos de visión	617	16369	940	12	6
Algoritmo de control	27845	57533	1451	4	98
Otros	1972	6175	81	0	0
Total	30434	80077	2472	16	104
Ocupación FPGA	59 %	37.6%	3%	1.3%	9%

de la FPGA Kintex 7 de Xilinx usada en esta Tesis. El programa de la FPGA se ha optimizado usando de nuevo un cauce segmentado y compartiendo núcleos IP entre los diferentes módulos. A través del interfaz proporcionado, el usuario es capaz de acceder a múltiples controladores visuales directos, facilitando así el estudio de las nuevas propuestas al poder compararlos sin necesidad de recompilar el programa de nuevo en la FPGA. Si comparamos la ocupación de los recursos de la FPGA en este caso, con la ocupación de recursos mostrados en la Tabla 5.4, vemos que el porcentaje de puertas lógicas casi se ha triplicado. También se ha ampliado el uso del resto de recursos. El precio pagado por disponer de un controlador más flexible ha sido un número mayor de recursos gastados en la FPGA. Sin embargo, el sistema sigue teniendo al menos un 40% de puertas lógicas disponibles, más del 60% de LUTs y más del 90% del resto de recursos.

La Figura 5.20 muestra el cauce segmentado del sistema implementado en la FPGA. La ley de control descrita en la Ecuación (5.18) se ha implementado en la FPGA tal y como se describe en la Figura 5.6. Este módulo contiene una gran cantidad de cálculos como multiplicación de matrices, inversión de matrices, etc. Sin embargo, el cálculo más complejo en este módulo es la raíz cuadrada de la matriz **W**. Para ello, como se ha explicado en el Apartado 5.3, se utiliza el algoritmo descrito en [Deadman-2013]. Como se observa en la Figura 5.20, el algoritmo de control empieza antes de la obtención del centro de gravedad del objeto. Esto es así porque algunos de los términos de la ley de control (5.18) no dependen de la información visual, como por ejemplo las matrices **M**, **C**, **g**, etc., que únicamente dependen de la dinámica del robot y se pueden calcular por lo tanto antes de que el procesamiento del fotograma actual acabe. Esto permite reducir la latencia

del sistema completo. Es importante mencionar en este punto que las matrices **M**, **C**, **g**, $\mathbf{L}_j$ y sus derivadas se calculan siguiendo un método analítico similar al descrito en [Kelly-2000]. Aprovechando las capacidades de paralelización de las FPGAs, los términos independientes se calculan al mismo tiempo, mientras que los que dependen del cálculo previo de otros términos tienen que esperar a que las entradas necesarias estén preparadas. Por ejemplo, en la implementación realizada, todos los elementos de las matrices (**M**, **C**, **g**, etc.) se calculan simultáneamente, explotando al máximo las capacidades de ejecución paralela (paralelismo espacial) de los dispositivos FPGA. Así, si se cambia el número de grados de libertad por cambio del robot empleado (pasar por ejemplo de 3 gdl a 6 gdl), el tiempo de cálculo del algoritmo de control será mayor, incrementando exponencialmente la ocupación de los recursos hardware de la FPGA. Sin embargo, se sigue un método de optimización basado en compartir los núcleos inactivos entre varias operaciones, o la ejecución de éstas en un cauce segmentado. Finalmente, la Figura 5.20 muestra también la frecuencia de actualización del sistema de control. Como muestra la gráfica, este tiempo depende del tiempo de adquisición de la cámara empleada para los experimentos. Así, el tiempo de adquisición de imagen es el cuello de botella del sistema. El paralelismo permite obtener un nuevo conjunto de pares articulares para el robot en sólo 2.9 ms. Ésta es la frecuencia de actualización que define el tiempo real del sistema propuesto. Y esta es una de las principales aportaciones de los sistemas de control visual directo embebidos en FPGA propuestos en esta Tesis. El sistema de control visual directo se convierte en un sistema de tiempo real, proporcionando una respuesta en este caso en 2.9 ms. Aunque hay que reseñar aquí que el uso de una cámara capaz de adquirir las imágenes a mayor frecuencia permitiría reducir este tiempo.

5.5.2 Control visual del robot Mitsubishi PA-10 de 7 grados de libertad

Esta sección describe dos experimentos en los que el controlador visual es integrado en la arquitectura propuesta basada en FPGA para el guiado del robot Mitsubishi PA-10. En la Figura 5.21 se muestra el montaje experimental empleado en esta sección. Se observa el robot con la cámara en su extremo, así como el objeto desde el cual se extraen las características visuales.

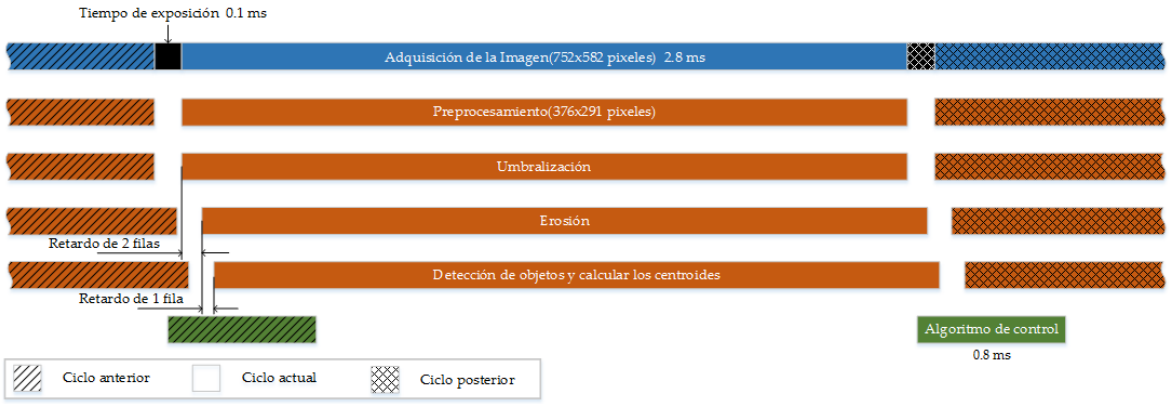


Figura 5.20 Experimento 5: Diagrama de tiempos de la ejecución del sistema de control visual con el framework mostrado en (5.18) en el robot COOPER

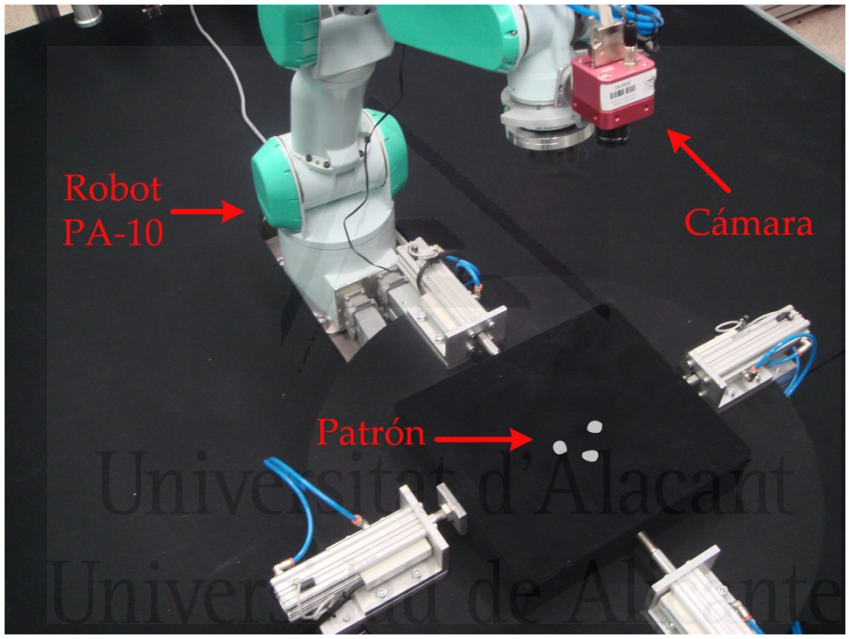
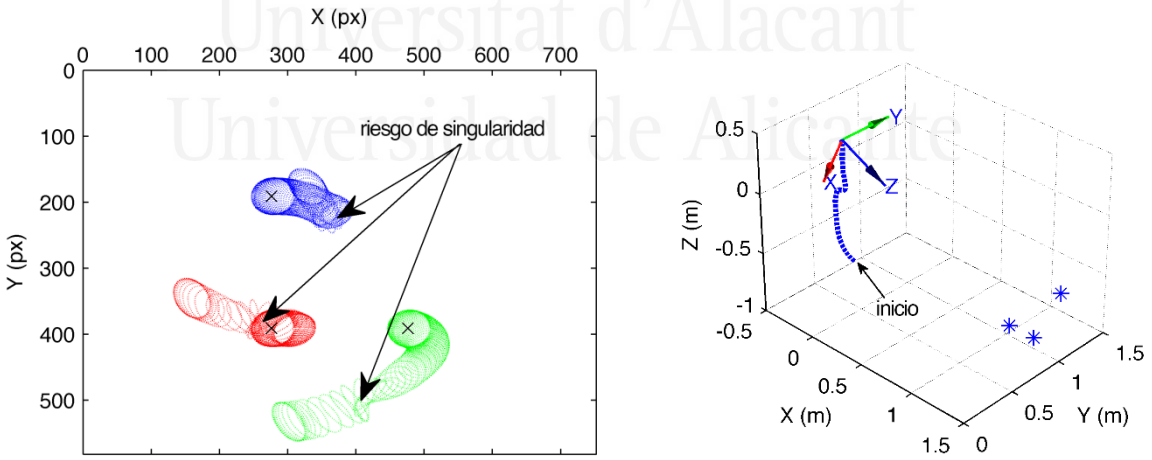


Figura 5.21 Montaje experimental para la evaluación de la arquitectura en el robot Mitsubishi PA-10.

5.5.2.1 Experimento 1

En este experimento se emplean 3 puntos característicos en la imagen para realizar el guiado del robot. El valor de las características iniciales extraídas en la imagen son $\mathbf{s} = [f_1 = (328, 175), f_2 = (159, 345), f_3 = (301, 532)]$ px y las finales $\mathbf{s}_d = [f_{1d} = (276, 191), f_{2d} = (276, 391), f_{3d} = (476, 391)]$ px. Al inicio del experimento, el extremo del robot se localiza en (0.5963, -0.1501, -0.01435) m. Usando el controlador propuesto en (5.8) se obtienen las trayectorias en la imagen representadas en la Figura 5.22.a para cada característica. Esta figura

representa, no sólo la evolución de las características en la imagen, sino también el valor de la perceptibilidad dinámica en cada momento [PérezJ-2016]. La perceptibilidad dinámica, representada en forma de elipses, representa la capacidad de movimiento de cada punto en la imagen cuando se usa control visual. El concepto de perceptibilidad dinámica permite integrar información dinámica del robot de manera que las elipses indican qué direcciones de movimiento son más factibles atendiendo a la cinemática y dinámica del robot, así como la configuración de las características en la imagen en cada momento. La forma de estas elipses permite tener información, por ejemplo, de la cercanía del robot a una configuración singular, tanto del robot, como en la imagen. Como se observa en la trayectoria descrita por las características en la imagen en la Figura 5.22.a, las elipses presentan una forma irregular cuando la trayectoria pasa cerca de una singularidad. Esta zona se indica en la figura como "riesgo de singularidad", y en esa zona se obtiene una manipulabilidad de 0.004, indicando que el robot se encuentra cerca de una singularidad. En cualquier caso, el robot alcanza la configuración deseada para las características en la imagen. En la Figura 5.22.b se representa la trayectoria Cartesiana 3D descrita por el robot hasta alcanzar la posición final deseada.



(a) (b)

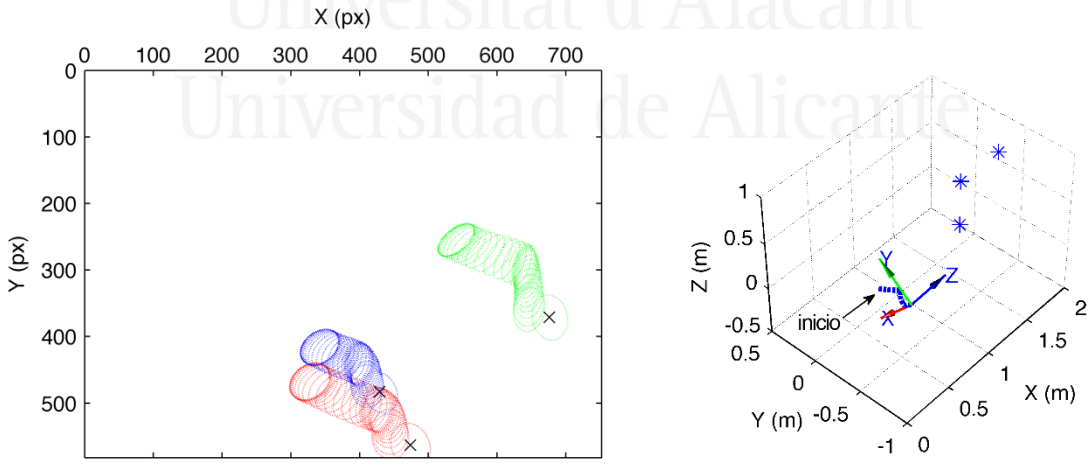
Figura 5.22 Experimento 1: Robot Mitsubishi PA-10. **(a)** Evolución de las características en la imagen y perceptibilidad dinámica. **(b)** Evolución de la posición del efector final del robot en el espacio Cartesiano 3D

5.5.2.2 Experimento 2

En este experimento de nuevo se considera un total de tres puntos como características en la imagen empleadas para realizar el guiado del robot. El valor de las características iniciales captadas por el sistema de visión son $\mathbf{s} = [f_1 = (341, 415), f_2 = (327, 467), f_3 = (540, 254)]$ px y las características finales $\mathbf{s}_d = [f_{1d} = (429, 483), f_{2d} = (474, 563), f_{3d} = (676, 371)]$ px. Como en el experimento anterior, en la Figura 5.23 se representa la trayectoria en la imagen, así como la trayectoria Cartesiana 3D descrita por el extremo del robot durante el experimento. La posición final deseada para las características en la imagen se ha marcado con una X. Como se observa en las elipses de perceptibilidad dinámica, el robot pasa cercano a una singularidad durante la trayectoria. En cualquier caso, se alcanza la posición final deseada tanto en la imagen como en 3D.

5.6 Conclusiones

En este capítulo se ha propuesto un controlador visual directo basado en imagen para el seguimiento de trayectorias, así como su implementación en la arquitectura basada en FPGA descrita en el Capítulo 3. Se ha comparado su comportamiento frente a una implementación basada en FPGA de un controlador basado en Jacobiana transpuesta



(a) (b)

Figura 5.23 Experimento 2: Robot Mitsubishi PA-10. **(a)** Evolución de las características en la imagen y perceptibilidad dinámica. **(b)** Evolución de la posición del efector final del robot en el espacio Cartesiano 3D

para posicionamientos puros. Se observa que se obtiene un mejor comportamiento presentando menores oscilaciones. A lo largo del capítulo se ha detallado distintos aspectos de su implementación así como la optimización de recursos de la FPGA. También se ha integrado en la arquitectura semiabierta propuesta en el Capítulo 3 un framework que permite en tiempo de ejecución definir una matriz **W** que proporciona distintos comportamientos dinámicos del sistema.

Una vez definido el controlador basado en imagen propuesto, se ha definido e integrado en el controlador un compensador de caos. El principal objetivo de este compensador es la obtención de un movimiento articular periódico cuando el extremo sigue una trayectoria repetitiva en el espacio imagen. Una de las ventajas de la implementación basada en FPGA no es sólo la reducción de tiempos y la ejecución paralela de distintos módulos, sino también la posibilidad de obtener retardos constantes y predecibles, convirtiendo al sistema en un sistema de tiempo real. Este aspecto ha permitido mejorar el desempeño del compensador de caos frente a implementaciones clásicas.

6 Conclusiones

En este último Capítulo se describen las conclusiones más destacadas acerca de las investigaciones desarrolladas y presentadas a lo largo de la presente Tesis. Los principales resultados alcanzados con estas investigaciones se han ido publicando en distintos medios científicos. En el Capítulo se describen brevemente las principales aportaciones realizadas en cada uno de ellos. Por último, se ofrecen diversas ideas para mejorar los distintos esquemas de control propuestos. Se exponen los distintos trabajos futuros que permitirán extender los sistemas de control visual directo a tareas de manipulación en las que están implicadas dos robots manipuladores dotados con sendas manos robóticas a su vez dotadas con sensores táctiles.

6.1 Conclusiones

Las investigaciones presentadas a lo largo de la Tesis se han centrado en un objetivo fundamental, y a la vez crítico, en los sistemas de control visual directo como es la mejora del desempeño de su funcionamiento mediante el uso de controladores basados en FPGAs. Para alcanzarlo ha sido necesario abordar investigaciones previas que a su vez han constituido las distintas aportaciones científicas que conforman la presente Tesis. En primer lugar, ha sido necesario definir, diseñar, construir, implementar y programar la arquitectura hardware necesaria para dar soporte a los controladores propuestos. Así, en el Capítulo 3 se ha descrito dicha arquitectura basada en FPGA, en la que se presta especial atención a la reducción en los tiempos de cadencia, procesamiento paralelo, e incluso en la posibilidad de aplicarse a distinto tipo de robots. La flexibilidad de la arquitectura diseñada y la reconfigurabilidad de las FPGAs es clave en este aspecto.

Los sistemas de control visual directo requieren realizar la extracción de características visuales necesarias para realizar el guiado del manipulador. Como se ha comprobado a lo largo de la Tesis, la mayoría de los retardos en estos sistemas proceden de los sistemas de visión artificial empleados (tanto en el proceso de captura como en el procesamiento necesario para extraer la información útil de la imagen). En el Capítulo 4 se

han descrito los algoritmos de visión artificial así como el procesamiento paralelo basado en FPGAs propuesto para obtener dicha información visual. Se propone el diseño de un cauce segmentado, que permite ejecutar en tiempo real el flujo de datos proporcionado por el digitalizador. De esta manera, la FPGA realiza el procesamiento de cada píxel para segmentar la imagen y etiquetar los objetos de la escena y posteriormente obtener su centroide. Se comprueba que el cauce segmentado permite trabajar a la frecuencia máxima de la cámara reduciendo así al máximo la necesidad de uso de memoria en la FPGA.

Una vez definida la arquitectura y los algoritmos de visión artificial, queda implementar el núcleo fundamental de los sistemas de control visual como es el controlador directo. Así, se ha definido un nuevo sistema de control visual directo basado en imagen para el seguimiento de trayectorias dependientes del tiempo. Estos controladores emplean la información visual para generar los pares articulares requeridos, con el objetivo de conseguir el seguimiento de la trayectoria deseada en el espacio imagen. Una vez definido este controlador se propone una implementación de los distintos componentes del controlador basada en FPGA. Debido a la optimización de recursos de la FPGA llevada a cabo tal y como se describe en el Capítulo 3, es posible realizar la implementación en la FPGA de distintos controladores visuales. De esta manera, se han implementado en la FPGA otros controladores visuales directos clásicos como el basado en Jacobiana transpuesta u otros basados en control óptimo que permiten ajustar su precisión dependiendo de la función de peso seleccionada. Esto permite, no sólo comparar el comportamiento de los distintos controladores, sino también evaluar la arquitectura y la mejora obtenida al usar una implementación basada en FPGA. Para validar los distintos controladores implementados sobre la FPGA y poder comparar sus comportamientos con respecto a sistemas implementados sobre GPPs, se emplean dos robots manipuladores: COOPER, un robot planar de 3 gdl; y el Mitsubishi PA10, un robot antropomórfico de 7 gdl. Los resultados obtenidos permiten verificar tanto la disminución en los retardos como la obtención de tiempos de latencia uniformes y predecibles.

6.2 Publicaciones

Seguidamente se citan las distintas publicaciones generadas a partir de las investigaciones presentadas en esta Tesis, desglosadas entre 3 artículos en revistas impactadas (primer cuartil), 2 aportaciones a congresos internacionales y 5 a congresos nacionales.

6.2.1 Revistas impactadas

- **"FPGA-based architecture for direct visual control robotic systems".** Alabdo, A.; Pérez, J.; García, G.J.; Pomares, J.; Torres, F., Mechatronics, 39, pp. 204-216, 2016.

  Este artículo describe el framework para desarrollar rápidamente sistemas de control visual dinámicos embebidos en una FPGA descrito más detalladamente en esta Tesis entre los capítulos 3, 4 y 5. El diseño paralelo de algoritmos incrementa la precisión de este tipo de controlador, mientras que minimiza el tiempo de respuesta. Los controladores visuales directos basados en imagen derivados de este framework permiten el seguimiento de trayectorias obteniendo diferentes comportamientos dinámicos dependiendo de una matriz de ponderación. También se añade a los controladores derivados de la matriz de ponderación un nuevo método para compensar el comportamiento caótico de este tipo de sistemas. Para comprobar la viabilidad del framework propuesto basado en FPGA, y para demostrar los efectos de las métricas, algunos de los controladores derivados se evalúan durante el seguimiento de trayectorias de imágenes. Este desarrollo no sólo supone la obtención del primer controlador visual directo sobre FPGA, sino que, aprovechando las características de estos dispositivos reconfigurables, permite convertir el sistema de control en un sistema de tiempo real, permitiendo obtener un ciclo de control estable y acotado en el tiempo. El artículo muestra las principales aportaciones y los resultados experimentales finales del proyecto público para grupos emergentes de la Universidad de Alicante "Control visual embebido de robots manipuladores

utilizando hardware reconfigurable FPGA" (GRE12-17), del cual fui investigador colaborador.

Este artículo ha sido publicado recientemente en la revista 'Mechatronics', en noviembre de 2016. En 2015 (último año con datos hasta la fecha), la revista citada tenía un factor de impacto en el 'Journal Citation Reports' del 'ISI Web of Knowledge' de 1.871, y se situaba en la posición 31 de un total de 132 en la categoría de 'Engineering, Mechanical', estando por tanto en el primer cuartil. El índice de impacto de la revista 'Mechatronics' se ha mantenido estable desde 2013, año en que tenía un impacto de 1.823, siempre en ascenso desde 2010 (0.994).

- **"FPGA-based visual control system using dynamic perceptibility".** Perez, J.; Alabdo, A.; Pomares, J.; Garcia, G. J.; Torres, F., Robotics and Computer-Integrated Manufacturing, 41 pp. 13-22, 2016.

  Esta publicación presenta el concepto de perceptibilidad dinámica, que proporciona información sobre la capacidad del sistema para realizar un seguimiento de los objetos mediante control visual directo. Este concepto se integra en los controladores propuestos y se implementa en la arquitectura integrada para evitar singularidades tanto en el espacio del robot como en la imagen. Para ello, se presenta un framework de control óptimo de robots mediante control visual directo que integra el concepto de perceptibilidad dinámica. Este artículo presenta la formulación del framework óptimo y detalla la implementación de uno de los controladores derivados en la arquitectura basada en FPGA descrita en la Tesis. En el artículo se describe cómo el framework implementado en FPGA para el desarrollo de controladores visuales directos se puede reconfigurar de manera muy sencilla para probar nuevos controladores o simplemente para agregar funcionalidades a los ya implementados. Bajo la misma base desarrollada en el artículo publicado en la revista Mechatronics, en este artículo se agrega al controlador un nuevo término que permite tener en cuenta la perceptibilidad dinámica. El poco tiempo requerido para implementar este nuevo controlador representa un gran avance en el diseño y pruebas de nuevos

controladores visuales directos embebidos en un sistema con características de tiempo real gracias a la FPGA.

Este artículo ha sido publicado recientemente en la revista 'Robotics and Computer-Integrated Manufacturing' en octubre de 2016. En 2015, la revista citada tenía un factor de impacto en el 'Journal Citation Reports' del 'ISI Web of Knowledge' de 2.077, y se situaba en la posición 9 de un total de 42 en la categoría de 'Engineering, Manufacturing', estando por tanto en el primer cuartil. También se situaba en 2015 en el primer cuartil en la categoría de 'Robotics', donde estaba situada en la posición 5 de un total de 25. El índice de impacto de la revista 'Robotics and Computer-Integrated Manufacturing' se ha incrementado desde 2011, año en que tenía un impacto de 1.173.

- **"A Survey on FPGA-Based Sensor Systems: Towards Intelligent and Reconfigurable Low-Power Sensors for Computer Vision, Control and Signal Processing".** García, G. J., Jara, C. A., Pomares, J., Alabdo, A., Poggi, L. M., Torres, F., Sensors, 14(4) pp. 6247-6278, 2014.

Este artículo se trata de un artículo de revisión. Uno de los primeros trabajos realizados en el marco del proyecto de la Universidad de Alicante para grupos emergentes GRE12-17 fue un estudio de los sistemas de control y visión artificial embebidos en una FPGA. El uso de FPGAs proporciona una tecnología específica de hardware reprogramable que permite obtener un sistema de sensores reconfigurable. Esta capacidad de adaptación permite la implementación de aplicaciones complejas utilizando la reconfigurabilidad parcial con un consumo de muy baja potencia. En este artículo se presentaba una revisión de los desarrollos de control, sensorización y visión artificial sobre FPGA, describiendo también las tecnologías FPGA empleadas por los diferentes grupos de investigación y proporcionando una visión general de la investigación futura en este campo. El estado del arte permitió centrar las investigaciones que se desarrollarían en el proyecto, y permitió detectar las necesidades no cubiertas en el campo. En especial, el artículo muestra cómo el uso de FPGAs se ha ido incrementando en los

últimos años para implementar controladores reconfigurables de robots. Sin embargo, son pocos los sistemas desarrollados en el campo del control visual, detectando además la inexistencia de sistemas de control visual directo implementados sobre una FPGA.

Este artículo fue publicado en la revista 'Sensors' en abril de 2014. En 2014, la revista citada tenía un factor de impacto en el 'Journal Citation Reports' del 'ISI Web of Knowledge' de 2.245, y se situaba en la posición 10 de un total de 56 en la categoría de 'Instruments & Instrumentation', estando por tanto en el primer cuartil. El índice de impacto de la revista 'Sensors' en 2014 es el mayor índice de impacto de la revista desde que se creó en 2004, año en que tenía un impacto de 0.990. Desde su publicación, el artículo ha sido citado un total de 7 veces según el 'ISI Web of Knowledge' y 15 veces según Google Schoolar.

6.2.2 Congresos Internacionales

- **"FPGA-based visual control of robot manipulators using dynamic perceptibility"**, J. Pérez, A. Alabdo, G. J. García, J. Pomares, F. Torres., En Proceedings of the International Conference on ReConFigurable Computing and FPGAs (ReConFig). Cancun, Mexico. 2015.

  En este artículo se describe el controlador propuesto basado en dinámica inversa para el seguimiento de trayectorias así como su implementación en la FPGA. Además, se describe el concepto de perceptibilidad dinámica como un criterio para detectar no sólo singularidades del robot sino también singularidades en la imagen. Ambas aportaciones se integran con el objetivo de permitir el seguimiento de trayectorias en la imagen evitando posibles singularidades que aparezcan durante dicho seguimiento.

- **"FPGA-based Framework for Dynamic Visual Servoing of Robot Manipulators",** A. Alabdo, J. Pérez, J. Pomares, G.J. Garcia, F. Torres., En Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation. Luxemburgo. 2015.

Este artículo describe el framework basado en control óptimo para el guiado de robots manipuladores. Dicho framework permite generar leyes de control visual directo basado en imagen con distinto comportamiento dependiendo de la función de peso empleada. Tras describir el framework y los distintos componentes de los controladores, se detalla su implementación en la arquitectura propuesta basada en FPGA.

6.2.3 Congresos Nacionales

- **"Control visual dinámico basado en FPGA de un robot manipulador de 6 grados de libertad",** A. Alabdo, J. Pérez, J. Pomares, G. J. García, F. Torres., En Proceedings de las XXXVII Jornadas de Automática, Madrid, Spain, Septiembre 2016.

  Este artículo describe los fundamentos de los distintos componentes hardware que conforman la arquitectura empleada para la implementación de los distintos controladores propuestos a lo largo de la Tesis. Se describen los principales elementos de un controlador visual directo basado en imagen y como se pueden implementar en la FPGA. En este artículo se aplica al guiado de un robot Mitsubishi PA-10 con 6 grados de libertad.

- **"ViSeC-Matlab: Una herramienta para el aprendizaje de sistemas de control visual sobre Matlab",** J. Pérez, A. Alabdo, G. J. García, J. Pomares, F. Torres., En Proceedings de las XXXVI Jornadas de Automática, Bilbao, Spain, Septiembre 2015.

  En este artículo se presenta la herramienta ViSeC desarrollada para evaluar mediante simulación los distintos controladores presentados a lo largo de la Tesis. Esta herramienta, basada en la toolbox de robótica de Matlab, permite introducir fácilmente controladores visuales y evaluar su comportamiento generando gráficas tales como el error en imagen, acciones de control, trayectoria del robot y en el espacio imagen, etc. También permite definir fácilmente la tarea de control visual facilitando de esta manera el proceso de depuración previo a la implementación.

- **"Arquitectura abierta de control visual directo sobre FPGA",** A. Alabdo , G. J. García, J. Pomares, F. Torres., En Proceedings de las XXXV Jornadas de Automática, Valencia, Spain, Septiembre 2014.

  En este artículo se describen los principales bloques que conforman la arquitectura hardware basada en FPGA descrita en el Capítulo 3. Esta arquitectura se ha definido como abierta desde el punto de vista que puede aplicarse a distinto tipo de robots sin más que implementar ciertos módulos que dependen del hardware controlador (pudiendo mantenerse por completo otros como es el control o el procesamiento visual).

- **"Control visual embebido en dispositivo FPGA",** Alacid, B.; Torres, A.; Alabdo, A.; García, G.J., En Actas de la Semana de la Automática y la Robótica de Alicante (SARA 2014), Alicante, Spain, Julio 2014; pp. 75-79.

  Este artículo describe las primeras implementaciones realizadas en una FPGA y que conformaron los inicios de la Tesis. Se resumen el comportamiento de los controladores basados en Jacobiana transpuesta, y se muestra un ejemplo de posicionamiento empleando una realimentación visual obtenida mediante la FPGA.

- **"Tracking de puntos para un sistema de control visual embebido en FPGA",** Torres, A.; Alacid, B.; Alabdo, A.; García, G.J., En Actas de la Semana de la Automática y la Robótica de Alicante (SARA 2014), Alicante, Spain, Julio 2014; pp. 81-85.

  Como se ha descrito a lo largo de la Tesis, la principal fuente de retardos en los sistemas de control visual es el proceso de captura y procesamiento de imagen. Los trabajos descritos en este artículo muestran cómo puede utilizarse la FPGA y su procesamiento paralelo con el objetivo de mejorar el proceso de extracción de características necesarias para los sistemas de control visual basados en imagen.

6.3 Trabajos futuros

Las investigaciones realizadas a lo largo de la Tesis han abierto líneas de investigación que podrían abordarse con la experiencia y desarrollos llevados a cabo. Dentro de estas líneas de investigación futuras cabe mencionar el diseño de sistemas de control visual embebidos para el guiado de manos robóticas. Actualmente, las investigaciones que han dado lugar a esta Tesis se están aplicando al proyecto financiado por el Ministerio de Economía y Competitividad y fondos FEDER "Sistema robótico multisensorial con manipulación dual para tareas asistenciales humano-robot" (DPI2015-68087-R). El objetivo fundamental de este proyecto es el desarrollo de un torso robótico multisensorial para su aplicación a tareas asistenciales humano-robot. Este torso robótico está compuesto por dos robots manipuladores dotados cada uno de ellos con sendas manos robóticas. Los desarrollos presentados a lo largo de la Tesis permiten realizar el guiado de brazos robots articulares empleando visión artificial. Sin embargo, los controladores implementados no pueden llevar a cabo el guiado de manos robóticas tal y como se ha descrito a lo largo de la Tesis. Estos controladores podrían extenderse incluyendo las características cinemáticas y dinámicas de estas manos robóticas. No existen experiencias previas de sistemas de control visual directos embebidos basados en FPGAs para el guiado de manos robóticas. Así, la experiencia obtenida en la Tesis abrirá en un futuro próximo un nuevo campo de investigación prometedor como es el guiado de sistemas de manipulación basados en manos robóticas empleando control visual directo. El uso de FPGAs permitirá no sólo reducir los tiempos de procesamiento sino también tener cadencias constantes y predecibles.

Otro campo de investigación será la extensión de los controladores visuales directos para el caso de control bimanual. En este caso ha de considerarse, no sólo el guiado de los manipuladores y las manos robóticas correspondientes, sino también la coordinación de movimientos entre las mismas. Además, cuando se realizan tareas de manipulación ha de considerarse no sólo la información visual procedente del entorno, sino también información táctil relativa a la fuerza de interacción que ejerce cada uno de los dedos sobre el objeto manipulado. La extensión de los controladores propuestos basados en

FPGAs para el control bimanual y considerando la información táctil será otro campo de investigación derivado de la Tesis en un futuro próximo.

En esta Tesis se ha considerado el uso de un sistema de visión artificial compuesto de una única cámara ubicada en el extremo del robot manipulador. El uso de un sistema de visión artificial en tareas de colaboración humano-robot a menudo provoca que la información proporcionada por esa única cámara no sea suficiente debido a la existencia de oclusiones. Se plantea como trabajo futuro el uso de distintos sistemas de visión artificial. De esta manera, los controladores desarrollados tendrán que hacer uso de los múltiples sistemas de visión ubicados en el espacio de trabajo para usar uno u otro dependiendo de cuál es el más conveniente para el posicionado del robot. Será necesario, no sólo definir un criterio para determinar qué sistema de visión es el más adecuado en cada momento, sino también el coordinar las acciones de control obtenidas a partir de cada sistema de visión con el objetivo de generar el movimiento final.

Referencias

[Abdul-Hafez-2014] Abdul-Hafez, A.H., Cervera, E. "Particle-filter-based pose estimation from controlled motion with application to visual servoing". *International Journal of Advanced Robotic Systems*, vol. 11, 2014.

[Acevedo-Avila-2016] Acevedo, R., Gonzalez, M., Garcia, A. "A Linked List-Based Algorithm for Blob Detection on Embedded Vision-Based Sensors", *Sensors* (Basel, Switzerland), vol. 16(6), 782, 2016.

[Adelson-1994] Adelson, E.H., Wang, J.Y.A., Niyogi, S.A. "Mid-level vision: new directions in vision and video". IEEE International Conference Image Processing, Austin, USA, vol. 2, pp. 21-25. 1994.

[Agarwal-2000] Agarwal, V., Hrishikesh, M.S., Keckler, S. W. Burger, D. "Clock rate versus ipc: The end of the road for conventional microarchitectures". 27th Annual International Symposium on Computer Architecture, 2000.

[Akkaya-2014] Akkaya, S., Akbatı, O., Görgün, H. "Multiple closed loop system control with digital PID controller using FPGA". IEEE International Conference on Control, Decision and Information Technologies, pp. 764-769, 2014.

[Alabdo-2014] Alabdo, A., Garcia, G.J., Pomares, J., Torres, F. "Arquitectura abierta de control visual directo sobre FPGA", XXXV Jornadas de Automática, Valencia, Spain, 2014.

[Alabdo-2015] Alabdo, A., Pérez, J., Pomares, J., Garcia, G.J., Torres, F. "FPGA-based framework for dynamic visual servoing of robot manipulators". IEEE International Conference on Emerging Technologies and Factory Automation, 2015.

[Alabdo-2016a] Alabdo, A., Pérez, J., Garcia, G.J., Pomares, J., Torres, F. "FPGA-based architecture for direct visual control robotic systems", *Mechatronics*, vol. 39, pp. 204-216. 2016.

[Alabdo-2016b] Alabdo, A., Pérez, J., Pomares, J., Garcia, G.J., Torres, F. "Control visual dinámico basado en FPGA de un robot manipulador de 6 grados de libertad", XXXVII Jornadas de Automática, Madrid, Spain, 2016.

[Alacid-2014] Alacid, B., Torres, A., Alabdo, A., García, G.J. "Control visual embebido en dispositivo FPGA", Semana de la Automática y la Robótica de Alicante, Alicante, Spain, pp. 75-79. 2014.

[Albo-Canals-2012] Albo-Canals, J., Ortega, S., Perdices, S., Badalov, A., Vilasis-Cardona, X. "Embedded low-power low-cost Camera Sensor based on FPGA and its applications in mobile robots". 19th IEEE International Conference on Electronics, Circuits, and Systems, Sevilla, Spain, pp. 336-339. 2012.

[Alcantara-2007] Alcantara, S., Pedrett, C., Vilanova, R., Moreno, R. "An undergraduate laboratory course on fuzzy controller implementation in FPGAs". Mediterranean Conference on Control & Automation, Athens, Greece, pp. 1-6. 2007.

[Ali-2013] Ali, N. A., Salim, S. I. M., Rahim, R. A., Anas, S. A., Noh, Z. M., Samsudin, S. I. "PWM controller design of a hexapod robot using FPGA". IEEE International Conference on Control System, Computing and Engineering, pp. 310-314, 2013.

[Aliakbarpour-2014] Aliakbarpour, H., Tahri, O., Araujo, H. "Visual servoing of mobile robots using non-central catadioptric cameras", *Robotics and Autonomous Systems*, vol. 62(11), pp. 1613-1622, 2014.

[Allen-1991] Allen, P. K., Yoshimi, B., Timcenko, A., "Real-Time Visual Servoing", IEEE International Conference on Robotics and Automation, Sacramento, CA, USA, pp. 851-856, 1991.

[Altera-2009] Altera, Nios II Processor Reference Handbook Ver. 9.1, 2009.

[Alvarez-2006] Alvarez, J., Lago, A., Nogueiras, A., Martinez-Penalver, C., Marcos, J., Doval, J., Lopez, O., "FPGA implementation of a fuzzy controller for automobile DC-DC converters". IEEE International Conference on Field Programmable Technology, Bangkok, Thailand, pp. 237-240, 2006.

[Ardilla-2016] Ardilla, F., Wibowo, I. K. "Design and implementation of fpga soft processor for holonomic robot low level control". IEEE Electronics Symposium pp. 197-202, 2016.

[Arias-Estrada-2001] Arias-Estrada, M., Torres-Huitzil, C. "Real-time field programmable gate array architecture for computer vision". *Journal of Electronic Imaging*, vol. 10, pp. 289-296, 2001.

[Astarloa-2009] Astarloa, A., Lazaro, J., Bidarte, U., Jimenez, J., Zuloaga, A. "FPGA technology for multi-axis control systems," *Mechatronics*, vol. 19(2), pp. 258–268, 2009.

[Ata-2014] Ata, A.A., Salman, M.S. "FPGA-based image processing system for Quality Control and Palletization applications". IEEE International Conference on Autonomous Robot Systems and Competitions, Espinho, Portugal, 2014.

[Bailey-2011] Bailey, D. G. "Design for Embedded Image Processing on FPGAs", John Wiley & Sons: Singapore, Singapore, 2011.

[Barranco-2012] Barranco, F., Diaz, J., Gibaldi, A., Sabatini, S.P., Ros, E. "Vector Disparity Sensor with Vergence Control for Active Vision Systems", *Sensors*, vol. 12, pp. 1771-1799, 2012.

[Batlle-2002] Batlle, J. A, "New FPGA/DSP-Based Parallel Architecture for Real-Time Image Processing". *Real-Time Imaging*, vol. 8, pp. 345–356, 2002.

[Baturone-2014] Baturone, I., Gersnoviez, A., Barriga, A. "Neuro-fuzzy techniques to optimize an FPGA embedded controller for robot navigation" *Applied Soft Computing Journal*, vol. 21, pp. 95-106, 2014.

[Besbes-2015] Besbes B., Rogozan A., Rus A.M., Bensrhair A., Broggi A. "Pedestrian Detection in Far-Infrared Daytime Images Using a Hierarchical Codebook of SURF". *Sensors*, vol. 15, pp. 8570–8594, 2015.

[Bochem-2011] Bochem, A., Kent, K. B., Herpers, R. "FPGA based real-time object detection approach with validation of precision and performance", IEEE International Symposium on Rapid System Prototyping, vol. 9(15), pp. 24-27, 2011.

[Bonabi-2014] Bonabi, S.Y., Asgharian, H., Safari, S., Ahmadabadi, M.N. "FPGA implementation of a biological neural network based on the hodgkin-huxley neuron model" Frontiers in Neuroscience, 2014.

[Botella-2011] Botella, G., Martín H., J.A., Santos, M., Meyer-Baese, U. "FPGA-Based Multimodal Embedded Sensor System Integrating Low- and Mid-Level Vision", *Sensors*, vol. 11, pp. 8164-8179, 2011.

[Bravo-2006] Bravo, I., Jimenéz, P., Mazo, M., Lázaro, J.L., Martín, E. "Architecture Based on FPGA's for Real-Time Image Processing", *Reconfigurable Computing: Architectures and Application*; Bertels, K., Cardoso, J.M.P., Vassiliadis, S., Eds.; Springer Berlin Heidelberg: Berlin, Germany, pp. 152-157, 2006.

[Bravo-2007] Bravo, I. "Arquitectura basada en FPGAs para la detección de objetos en movimiento, utilizando visión computacional y técnicas PCA". Doctoral Thesis, Universidad de Alcalá, Madrid, 2007.

[Bravo-2010] Bravo, I., Mazo, M., Lázaro, J.L., Gardel, A., Jiménez, P., Pizarro, D. "An Intelligent Architecture Based on Field Programmable Gate Arrays Designed to Detect Moving Objects by Using Principal Component Analysis", *Sensors*, vol. 10, pp. 9232-9251, 2010.

[Briot-2016] Briot, S., Chaumette, F., Martinet, P. "Revisiting the Determination of the Singularity Cases in the Visual Servoing of Image Points Through the Concept of Hidden Robot", IEEE Transactions on Robotics. Article in Press, 2017.

[Bruyninckx-2001] Bruyninckx, H., "Open robot control software: the orocos project". IEEE International Conference on Robotics and Automation, vol. 3, pp. 2523-2528, 2001.

[Cai-2013] Cai, C., Dean-Leon, E., Mendoza, D., Somani, N., Knoll, A. "Uncalibrated 3D stereo image-based dynamic visual servoing for robot manipulators" IEEE International Conference on Intelligent Robots and Systems, pp. 63-70, 2013.

[Calmon-2002] Calmon, F., Fathallah, M., Viverge, P.J., Gontrand, C., Carrabina, J., Foussier, P. "FPGA and Mixed FPGA-DSP Implementations of Electrical Drive

Algorithms". In *Field-Programmable Logic and Applications: Reconfigurable Computing Is Going Mainstream*; Glesner, M., Zipf, P., Renovell, M., Eds.; Springer Berlin Heidelberg: Berlin, Germany, vol. 2438, pp. 1144–1147, 2002.

[Calvo-Gallego-2012] Calvo-Gallego, E., Aldaya, A.C., Brox, P., Sánchez-Solano, S. "Real-time FPGA connected component labeling system". IEEE International Conference on Electronics, Circuits and Systems, Seville, Spain, pp. 593–596, 2012.

[Camacho-2000] Camacho, P., Coslado, F., González, M., Sandoval, F. "Adaptive Multiresilution Imager Based on FPGAs". European Signal Processing Conference, Tampere, Finland, 2000.

[Camplani-2013] Camplani M., Mantecon T., Salgado L. "Depth-Color Fusion Strategy for 3-D Scene Modeling With Kinect". IEEE Transactions Cybernetics, vol. 43, pp. 1560–1571, 2013.

[Caron-2006] Caron, F., Duflos, E., Pomorski, D., Vanheeghe, P. "GPS/IMU data fusion using multisensor Kalman filtering: introduction of contextual aspects". *Information fusion*, vol. 7(2), pp. 221-230, 2006.

[Cazy-2015] Cazy, N., Wieber, P.-B., Giordano, P.R., Chaumette, F. "Visual servoing when visual information is missing: Experimental comparison of visual feature prediction schemes", IEEE International Conference on Robotics and Automation, pp. 6031-6036, 2015.

[Cervera-2001] Cervera, E., Berry, F., Martinet, P., "Stereo Visual Servoing with a Single Point: A Comparative Study", IEEE International Conference on Advanced Robotics, Budapest, Hungary, pp. 213-218, 2001.

[Cervera-2003] Cervera, E., Berry, F., Martinet, P., "Stereo Visual Servoing with Oriented Blobs", International Conference on Advanced Robotics, Coimbra, Portugal, pp. 977-982, 2003.

[Cervera-2008] Cervera, E., Garcia-Aracil, N., Martinez, E., Nomdedeu, L., del Pobil, A. P., "Safety for a Robot Arm Moving Amidst Humans by Using Panoramic Vision",

IEEE International Conference on Robotics and Automation, Pasadena, CA, USA, pp. 2183-2188, 2008.

[Comport-2003] Comport, A., Pressigout, M., Marchand, E., Chaumette, F. "A Visual Servoing Control Law that is Robust to Image Outliers", IEEE International Conference on Intelligent Robots and Systems, vol. 1, pp. 492-497, 2003.

[Copot-2012] Copot, C., Lazar, C., Burlacu, A. "Predictive control of nonlinear visual servoing systems using image moments", *IET Control Theory and Applications*, vol. 6 (10), pp. 1486-1496, 2012.

[Corke-2001] Corke, P. I.; Hutchinson, S. A., "A New Partitioned Approach to Image-Based Visual Servo Control". *IEEE Transactions on Robotics and Automation*, vol. 17(4) pp. 507-515, 2001.

[Coslado-2004] Coslado, F.J., Camacho, P., González, M., Sandoval, F. "Hardware Architecture for Hierarchical Segmentation in Foveal Images". *International Journal of Imaging Systems and Technology*, vol. 14, pp. 153-166, 2004.

[Crookes-1999] Crookes D, Benkrid K. "FPGA implementation of image component labelling". SPIE 3844, Reconfigurable Technology: FPGAs for Computing and Applications. Boston, pp. 17-23, 1999.

[Chang-2006] Chang, T. N., Biao, C., Sriwilaijaroen, P., "Motion Control Firmware for High-Speed Robotic Systems," *IEEE Transactions on Industrial Electronics*, vol.53(5), pp.1713-1722, 2006

[Chaumette-1998] Chaumette, F., "Potential Problems of Stability and Convergence in Image-Based and Position-Based Visual Servoing". The Confluence of Vision and Control, D. J. Kriegman, Ed. Springer-Verlag: New York, pp. 66-78, 1998.

[Chaumette-2006] Chaumette, F., Hutchinson, S. "Visual Servo Control, Part I: Basic Approaches". *IEEE Robotics and Automation Magazine*, vol. 13(4), pp. 82-90, 2006.

[Chaumette-2007] Chaumette, F., Hutchinson, S. "Visual Servo Control, Part II: Advanced Approaches". *IEEE Robotics and Automation Magazine*, vol. 14(1), pp. 109-118, 2007.

[Cheah-2010] Cheah, C. C., Hou, S. P., Zhao, Y., & Slotine, J. J. E. "Adaptive vision and force tracking control for robots with constraint uncertainty". *IEEE/ASME Transactions on Mechatronics*, 15(3), pp. 389-399, 2010.

[Cheah-2012] Cheah, C. C., Liu, C., Slotine, J. J. E. "Adaptive Vision based Tracking Control of Robots with Uncertainty in Depth Information". IEEE International Conference on Robotics and Automation, Roma, pp. 2817-2822, 2007.

[ChenJ-2005] Chen, J., Dawson, D. M., Dixon, W. E., Behal, A., "Adaptive Homography-Based Visual Servo Tracking for a Fixed Camera Configuration with a Camera-in-Hand Extension". *IEEE Transactions on Control Systems Technology*, vol. 13(5) pp. 814-825, 2005.

[ChenYL-2011] Chen Y.L., Wu B.F., Huang H.Y., Fan C.J. "A Real-Time Vision System for Nighttime Vehicle Detection and Traffic Surveillance". *IEEE Transactions on Industrial Electronics*, vol. 58, pp. 2030–2044, 2011.

[ChenZP-2009] Chen, Z. P., Lii, N. Y., Wu, K., Liu, H., Xue, Z. X., Jin, M. H., Liu, Y. W., Fan, S. W., Lan, T. "Flexible FPGA-Based Controller Architecture for Five-fingered Dexterous Robot Hand with Effective Impedance Control". In Proc IEEE International Conference on Robotics and Biomimetics, pp. 1063 – 1068, 2009.

[Chesi-2005] Chesi, G., Prattichizzo, D., Vicino, A., "Visual Servoing: Reaching the Desired Location Following a Straight Line Via Polynomial Parameterizations", IEEE International Conference on Robotics and Automation, Barcelona, España, pp. 2559-2564, 2005.

[Chinnaaiah-2015] Chinnaaiah, M. C., Ambati, S., Yaddla, M., Sravanthi, J., Sanjay, D. "FPGA based robots hardware efficient scheme for Real time indoor environment with behavioural control". International Conference on Innovations in Information, Embedded and Communication Systems, pp. 1-5, 2015.

[Chinnaaiah-2016a] Chinnaiah, M. C., Dubey, S., Anusha, K., Vani, J., Vani, G. D. "A versatile path planning algorithm with behavioural control using FPGA based

robots". International Conference on Intelligent Systems and Control, pp. 1-6, 2016.

[Chinnaaiah-2016b] Chinnaaiah, M. C., Priyanka, G., Vani, G. D., Jyoti, M. A., Vennela, K. "A new approach: An FPGA based robot navigation for patrolling in service environment". International Conference on Research Advances in Integrated Navigation Systems, pp. 1-4, 2016.

[Dagnino-2016] Dagnino, G., Georgilas, I., Tarassoli, P., Atkins, R., Dogramadzi, S. "Design and real-time control of a robotic system for fracture manipulation". International Conference of the Engineering in Medicine and Biology Society, pp. 4865-4868, 2016.

[Dahmouche-2012] Dahmouche, R., Andreff, N., Mezouar, Y., Ait-Aider, O., Martinet, P. "Dynamic visual servoing from sequential regions of interest acquisition", *International Journal of Robotics Research*, vol. 31(4), pp. 520-537, 2012.

[David-2004] David, P., DeMenthon, D., Duraiswami, R., Samet, H., "Softposit: Simultaneous Pose and Correspondence Determination". *International Journal of Computer Vision*, vol. 59 pp. 259-284, 2004.

[De Garis-2000a] De Garis, H., Korkin, M. "CAM-Brain Machine (CBM): Real time evolution and update of a 75 million neuron FPGA-based artificial brain" *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, vol. 24(2), pp. 241-262, 2000.

[De Garis-2000b] De Garis, H., Korkin, M., Gers, F., Nawa, E., Hough, M. "Building an artificial brain using an FPGA based CAM-Brain Machine" *Applied Mathematics and Computation*, vol. 111(2), pp. 163-192, 2000.

[De Garis-2001] De Garis, H., Korkin, M., Fehr, G. "The CAM-brain machine (CBM): An FPGA based tool for evolving a 75 million neuron artificial brain to control a lifesized kitten robot", *Autonomous Robots*, vol. 10(3), pp. 235-249, 2001.

[De Garis-2002] De Garis, H., Korkin, M. "The CAM-Brain Machine (CBM): An FPGA-based hardware tool that evolves a 1000 neuron-net circuit module in seconds and

updates a 75 million neuron artificial brain for real-time robot control" *Neurocomputing*, vol. 42, pp. 35-68, 2002.

[De Luca-2008] De Luca, A., Oriolo, G., Giordano, P. R., "Feature Depth Observation for Image-Based Visual Servoing: Theory and Experiments". *International Journal of Robotics Research*, vol. 27(10) pp. 1093-1116, 2008.

[Deadman-2013] Deadman, E., Higham, N. J., Ralha, R. "Blocked Schur Algorithms for Computing the Matrix Square Root", Revised Selected Papers Applied Parallel and Scientific Computing: 11th International Conference, pp. 171-182, 2013.

[Debenedictis-2016] Debenedictis, E.P. "It's Time to Redefine Moore's Law Again" *Computer*, vol. 50(2), pp. 72-75, 2017.

[Dementhon-1995] Dementhon, D. F., Davis, L. S., "Model-Based Object Pose in 25 Lines of Code". *International Journal of Computer Vision*, vol. 15(1-2) pp. 123-141, 1995.

[Deng-2003] Deng, L., Wilson, W.J., Janabi-Sharifi, F. "Dynamic Performance of the Position-Based Visual Servoing Method in the Cartesian and Image Spaces" IEEE International Conference on Intelligent Robots and Systems, vol. 1, pp. 510-515, 2003.

[Dhipa-2015] Dhipa, M., Jeyakkannan, N., Chandrasekar, A. "Hardware accelerated stereo imaging for visual servoing," Online International Conference on Green Engineering and Technologies (IC-GET), Coimbatore, pp. 1-4, 2015.

[Díaz-2004] Díaz, J.; Ros, E.; Mota, S.; Carrillo, R.; Agis, R. "Real Time Optical Flow Processing System". In *Field Programmable Logic and Application*; Becker, J., Platzner, M., Vernalde, S., Eds.; Springer Berlin Heidelberg, Berlin, Germany, vol. 3203, pp. 617–626, 2004.

[Díaz-2006] Díaz, J., Ros, E., Pelayo, F., Ortigosa, E.M., Mota, S. "FPGA-Based Real-Time Optical-Flow System". IEEE Transactions on Circuits Systems Video Technology, vol. 16, pp. 274-279, 2006.

[Doignon-2015] Doignon, C. "A single 3-D feature-based visual servoing" International Conference on Methods and Models in Automation and Robotics, pp. 41-46, 2015.

[Dominguez-2014] Domínguez, C., Hassan, H., Crespo, A., Albaladejo, J. "Multicore and FPGA implementations of emotional-based agent architectures" *Journal of Supercomputing*, vol. 71(2), pp. 479-507, 2014.

[Dominguez-2017] Dominguez, C., Hassan, H., Crespo, A. "Emotional robot control architecture implementation using FPGAs", *Journal of Systems Architecture*, vol. 72, pp. 29-41, 2017.

[Dubois-2008] Dubois, J., Ginhac, D., Paindavoine, M., Heyrman, B. "A 10 000 fps CMOS sensor with massively parallel image processing". *IEEE Journal of Solid-State Circuits*, vol. 43, pp. 706-717, 2008.

[Eum-2013] Eum S., Jung H.G. "Enhancing Light Blob Detection for Intelligent Headlight Control Using Lane Detection". *IEEE Transactions Intelligent Transportation Systems*, vol. 14, pp. 1003–1011, 2013.

[Fang-2002] Fang, Y., Behal, A., Dixon, W. E., Dawson, D. M. "Adaptive 2.5D visual servoing of kinematically redundant robot manipulators," IEEE Conference on Decision and Control, vol.3, pp. 2860-2865, 2002.

[Fiack-2015] Fiack, L., Cuperlier, N., Miramond, B. "Embedded and real-time architecture for bio-inspired vision-based robot navigation, *Journal of Real-Time Image Processing*, vol.10(4), pp. 699-722, 2015.

[Fink-2015] Fink, G., Xie, H., Lynch, A.F., Jagersand, M. "Experimental validation of dynamic visual servoing for a quadrotor using a virtual camera" International Conference on Unmanned Aircraft Systems, pp. 1231-1240, 2015.

[Fons-2006] Fons, M., Fons, F., Cantó, E. "Design of FPGA-based Hardware Accelerators for On-line Fingerprint Matcher Systems". Proceedings of the Ph. D. Research in Microelectronics and Electronics, Otranto, Italy, pp. 333-336, 2006.

[Foxlin-2005] Foxlin, E. "Pedestrian tracking with shoe-mounted inertial sensors". *IEEE Computer graphics and applications*, vol. 25(6), pp. 38-46, 2005.

[Freeman-1989] Freeman, R. "Configurable Electrical Circuit Having Configurable Logic Elements and Configurable Interconnects", US Patent No. 4,870,302, 26 September 1989.

[Gadea-2000] Gadea, R., Cerdá, J., Ballester, F., Mocholí, A. "Artificial Neural Network Implementation on a single FPGA of a Pipelined OnLine Backpropagation", International symposium on System synthesis, Madrid, Spain, pp. 225-230, 2000.

[Gaisler-2007] Gaisler, J., Catovic, E., Habinc, S. *GRLIBIPLibraryUser'sManual*. Gaisler Research, 2007.

[Gamazo-2010] Gamazo-Real, J.C., Vázquez-Sánchez, E., Gómez-Gil, J. "Position and speed control of brushless DC motors using sensorless techniques and application trends". *Sensors*, vol. 10, pp. 6901-6947, 2010.

[GarciaGJ-2004] Garcia, G. J., Pomares, J., Torres, F., "Control Visual De Robots Manipuladores. Una Herramienta Para Su Diseño Y Aprendizaje", XXV Jornadas de Automática, Ciudad Real, España, pp. 120-127, 2004.

[GarciaGJ-2009a] Garcia, G. J., Corrales, J. A., Pomares, J., Torres, F., "Survey of Visual and Force/Tactile Control of Robots for Physical Interaction in Spain". *Sensors*, vol. 9(12), pp. 9689-9733, 2009.

[GarciaGJ-2009b] Garcia, G. J., Pomares, J., Torres, F., "Automatic Robotic Tasks in Unstructured Environments Using an Image Path Tracker". *Control Engineering Practice*, vol. 17(5) pp. 597-608, 2009.

[GarciaGJ-2014] Garcia, G.J., Jara, C.A., Pomares, J., Alabdo, A., Poggi, L.M., Torres, F. "A survey on FPGA-based sensor systems: Towards intelligent and reconfigurable low-power sensors for computer vision, control and signal processing" *Sensors*, vol. 14 (4), pp. 6247-6278, 2014.

[GarciaN-2011] Garcia-Aracil, N., Perez-Vidal, C., Sabater, J. M., Morales, R., Badesa, F. J. "Robust and Cooperative Image-Based Visual Servoing System Using a Redundant Architecture", *Sensors*, vol. 11(12), pp. 11885–11900, 2011.

[Geier-2016] Geier, M., Pitzl, F., Chakraborty, S. "GigE Vision Data Acquisition for Visual Servoing using SG/DMA Proxying". ACM/IEEE Symposium on Embedded Systems for Real-Time Multimedia, ACM, New York, NY, USA, pp. 19-28, 2016.

[Gil-2004] Gil, A., Gutiérrez, R., Alonso, J.L., Ávila, S.F. "Stereo Calculation of significant points using a FPGA". WSEAS International Conference on Electroscience and Technology for Naval and All-electric Ship, Athens, Greece, 2004.

[Gonzalez-2009] Gonzalez, C., Mozos, D., Resano, J. "FPGA support for satellite computations of hyper spectral images. In Proceedings of the International Conference on Field Programmable Logic and Applications", Prague, Czech Republic, pp. 715-716, 2009.

[González-2010] González, C., Resano, J., Mozos, D., Plaza, A., Valencia, D. "FPGA Implementation of the Pixel Purity Index Algorithm for Remotely Sensed Hyperspectral Image Analysis". *EURASIP Journal on Advances in Signal Processing* 2010.

[González-2012a] González, C., Mozos, D., Resano, J., Plaza, A. "FPGA Implementation of the N-FINDR Algorithm for Remotely Sensed Hyperspectral Image Analysis". *IEEE Transactions on Geoscience and Remote Sensing,* vol. 50, pp. 374-388, 2012.

[González-2012b] González, C., Resano, J., Plaza, A., Member, S., Mozos, D. "FPGA Implementation of Abundance Estimation for Spectral Unmixing of Hyperspectral Data Using the Image Space Reconstruction Algorithm". *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing,* vol. 5, pp. 248-261, 2012.

[González-2013] González, C., Sánchez, S., Paz, A., Resano, J., Mozos, D., Plaza, A. "Use of FPGA or GPU-based architectures for remotely sensed hyperspectral image processing". *Integration, the VLSI Journal*, vol. 46, pp. 89-103, 2013.

[Gürsoy-2016] Gürsoy, H., Efe, M.Ö. "Control System Implementation on an FPGA Platform" *IFAC-PapersOnLine*, vol.49 (25), pp. 425-430, 2016.

[Guzmán-2004] Guzmán, A., Beltrán, M. "Satellite On-Board Image Compression Adviser". IEEE International Symposium on Signal Processing and Information Technology, Rome, Italy, pp. 296-301, 2004.

[Hadj-Abdelkader-2008] Hadj-Abdelkader, H., Mezouar, Y., Martinet, P., Chaumette, F., "Catadioptric Visual Servoing from 3-D Straight Lines". *IEEE Transactions on Robotics*, vol. 24(3) pp. 652-665, 2008.

[Hassan-2010] Hassan, S., Anwer, N., Khattak, Z., & Yoon, J. "Open architecture dynamic manipulator design philosophy (DMD)". *Robotics and Computer-Integrated Manufacturing*, vol. 26(2), pp. 156–161, 2010.

[Hedge-2013] Hegde G., Ye C., Robinson C., Stroupe A., Tunstel E. "Computer-Vision-Based Wheel Sinkage Estimation for Robot Navigation on Lunar Terrain". *IEEE/ASME Transactions Mechatronics*, vol. 18, pp. 1346–1356, 2013.

[Hentati-2014] Hentati, M., Aoudni, Y., Abid, M. "The implementation of basic morphological operations on FPGA using partial reconfiguration". International Image Processing, Applications and Systems Conference Sfax, Tunisia, 2014.

[Horaud-1997] Horaud, R., Dornaika, F., Lamiroy, B., "Object Pose: The Link between Weak Perspective, Paraperspective, and Full Perspective". *International Journal of Computer Vision*, vol. 22(2) pp. 173-189, 1997.

[HuJS-2011] Hu, J. S., Chang, N Y. C., Yang, J. J., Wang, J. J., Lossio, R. G., Chien, M. C., Chang, Y. J., Kai, C. Y., Su, S. H. "FPGA-based Embedded Visual Servoing Platform for Quick Response Visual Servoing" 8th Asian Control Conference, 2011.

[HuX-2013] Hu, X., Xie, K., Peng, T. "Research on direct calibration method of eye-to-hand system of robot" Proceedings of SPIE - The International Society for Optical Engineering, vol. 8916, 2013.

[HuangJB-2008] Huang, J.B., Xie, Z.W., Liu, H., Sun, K., Liu, Y.C., Jiang, Z.N. "DSP/FPGA-based controller architecture for flexible joint robot with enhanced

impedance performance" *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 53(3), pp. 247-261, 2008.

[HuangHC-2009] Huang, H.-C., Tsai, C.-C. "FPGA implementation of an embedded robust adaptive controller for autonomous omnidirectional mobile platform" *IEEE Transactions on Industrial Electronics*, vol. 56(5), pp. 1604-1616, 2009.

[HuangHC-2015] Huang, H.-C. "A Taguchi-Based Heterogeneous Parallel Metaheuristic ACO-PSO and Its FPGA Realization to Optimal Polar-Space Locomotion Control of Four-Wheeled Redundant Mobile Robots" *IEEE Transactions on Industrial Informatics*, vol. 11(4), pp. 915-922, 2015.

[HuangHC-2016a] Huang, H.-C., Chiang, C.-H. "An Evolutionary Radial Basis Function Neural Network with Robust Genetic-Based Immunecomputing for Online Tracking Control of Autonomous Robots" *Neural Processing Letters*, vol. 44(1), pp. 19-35, 2016.

[HuangHC-2016b] Huang, H.-C. "Fusion of Modified Bat Algorithm Soft Computing and Dynamic Model Hard Computing to Online Self-Adaptive Fuzzy Control of Autonomous Mobile Robots" *IEEE Transactions on Industrial Informatics*, vol. 12(3), pp. 972-979, 2016.

[HuangHC-2016c] Huang, H.-C., Chiang, C.-H. "Backstepping Holonomic Tracking Control of Wheeled Robots Using an Evolutionary Fuzzy System with Qualified Ant Colony Optimization" *International Journal of Fuzzy Systems*, vol. 18(1), pp. 28-40, 2016.

[HuangSJ-2015] Huang, S.-J., Liu, S., Wu, C.-H. "Intelligent humanoid mobile robot with embedded control and stereo visual feedback" *Journal of Mechanical Science and Technology*, vol. 29(9), pp. 3919-3931, 2015.

[Ibarguren-2014] Ibarguren, A., Martínez-Otzeta, J.M., Maurtua, I. "Particle filtering for industrial 6DOF visual servoing" *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 74 (3-4), pp. 689-696, 2014.

[Ivchenko-2015] Ivchenko, V.D., Krug, P.G., Kurakov, M.V., Matyukhina, E.N., Pavelyev, S.A. "The applying of the hardware-based reconfiguration for autonomous control systems of space mobile robots" *Journal of Theoretical and Applied Information Technology*, vol. 82(1), pp. 1-12, 2015.

[Jabbari-2014] Jabbari Asl, H., Oriolo, G., Bolandi, H. "Output feedback image-based visual servoing control of an underactuated unmanned aerial vehicle" Proceedings of the Institution of Mechanical Engineers. Part I: Journal of Systems and Control Engineering, vol. 228(7), pp. 435-448, 2014.

[Jabeen-2015] Jabeen, S., Srinivasan, S.K., Shuja, S., Dubasi, M.A.L. "A Formal Verification Methodology for FPGA-Based Stepper Motor Control", *IEEE Embedded Systems Letters*, vol. 7(3), pp. 85-88, 2015.

[Jablonski-2004] Jablonski M, Gorgon M, "Handel-C implementation of classical component labelling algorithm". Euromicro Symposium on Digital System Design, Rennes, pp. 387-393, 2004.

[Janabi-Sharifi-2010] Janabi-Sharifi, F., Marey, M. "A Kalman-filter-based method for pose estimation in visual servoing, *IEEE Transactions on Robotics*, vol. 26(5), pp. 939-947, 2010.

[Janabi-Sharifi-2011] Janabi-Sharifi, F., Deng, L., Wilson, W.J. "Comparison of basic visual servoing methods", *IEEE/ASME Transactions on Mechatronics*, vol. 16(5), pp. 967-983, 2011.

[Jara-2014] Jara, C.A., Pomares, J., Candelas, F.A., Torres, F. "Control framework for dexterous manipulation using dynamic visual servoing and tactile sensors' feedback" *Sensors*, vol. 14(1), pp. 1787-1804, 2014.

[Jeon-2002] Jeon, J.W., Kim, Y.-K. "FPGA based acceleration and deceleration circuit for industrial robots and CNC machine tools", *Mechatronics*, vol. 12 (4), pp. 635-642, 2002.

[Jimenez-Fernandez-2009] Jimenez-Fernandez, A., Lujan-Martinez, C., Paz-Vicente, R., Linares-Barranco, A., Jimenez, G., Civit, A. "From Vision Sensor to Actuators,

Spike Based Robot Control through Address-Event-Representation". In *Bio-Inspired Systems: Computational and Ambient Intelligence*; Cabestany, J., Sandoval, F., Prieto, A., Corchado, J. M., Eds.; Springer Berlin Heidelberg: Berlin, Germany, vol. 5517, pp. 797-804, 2009.

[Jimenez-Fernandez-2012] Jimenez-Fernandez, A., Jimenez-Moreno, G., Linares-Barranco, A., Dominguez-Morales, M.J., Paz-Vicente, R., Civit-Balcells, A. "A Neuro-Inspired Spike-Based PID Motor Controller for Multi-Motor Robots with Low Cost FPGAs", *Sensors*, vol. 12, pp. 3831-3856, 2012.

[Johnston-2008] Johnston, C., Bailey, D. "FPGA implementation of a single pass connected components algorithm," IEEE International Symposium on Electronic Design, Test and Applications, pp. 228 –231, 2008.

[Jorstad-2008] Jorstad, A., Burlina, P., Wang, I. J., Lucarelli, D., DeMenthon, D., "Model-Based Pose Estimation by Consensus", International Conference on Intelligent Sensors, Sensor Networks and Information Processing, Sydney, Australia, pp. 569-574, 2008.

[Juan-Manuel-2016] Juan-Manuel, R.-A., Saul, T.-A., Jose-Emilio, V.-S., Marco-Antonio, A.-F. "FPGA embedded pd control of A 1 dof manipulator with a pneumatic actuator" *International Journal of Robotics and Automation*, vol. 31(3), pp. 176-186, 2016.

[Jung-2007] Jung, S., Kim, S.S. "Hardware implementation of a real-time neural network controller with a DSP and an FPGA for nonlinear systems" *IEEE Transactions on Industrial Electronics*, vol. 54(1), pp. 265-271, 2007.

[Jwu-Sheng-2011] Jwu-Sheng, H., Yen-Chung, N., Jwu-Jiun, Y., Jyun-Ji, W., Gondim, R., Ming-Chih, C., Yung-Jung, C., Chen-Yu, K., Shyh-Haur, S., "FPGA-based Embedded Visual Servoing Platform for Quick Response Visual Servoing" *Proc Asian Control*, pp. 263-268, 2011.

[Kalman-1960] Kalman, R. E. "A New Approach to Linear Filtering and Prediction Problems". *ASME. J. Basic Eng*, vol. 82(1), pp. 35-45, 1960.

[Kaminaga-2016] Kaminaga, H., Ko, T., Yorita, S., Sato, S., Masumura, R., Komagata, M., Nakamura, Y. "Enhancement of mechanical strength, computational power, and heat management for fieldwork humanoid robots". IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids), pp. 786-793, 2016.

[Kassas-2011] Kassas, Z. M. "Methodologies for Implementing FPGA-Based Control Systems", IFAC World Congress, Milano, Italy, 2011.

[Kazemi-2013] Kazemi, M., Gupta, K.K., Mehrandezh, M. "Randomized kinodynamic planning for robust visual servoing" *IEEE Transactions on Robotics*, vol. 29(5), pp. 1197-1211, 2013.

[Kelly-2000] Kelly, R., Carelli, R., Nasisi, O., Kuchen, B., Reyes, F. "Stable Visual Servoing of Camera-in-Hand Robotic Systems". *IEEE/ASME Transactions on Mechatronics*, vol. 5(1), pp. 39-48, 2000.

[Kelly-2008] Kelly, R., Monroy, C., Bugarin, E., Cervantes, I., lvarez-Ramirez, J. "On transpose Jacobian control for monocular fixed-camera 3D Direct Visual Servoing". INTECH Open Access Publisher, 2008.

[Keshmiri-2017] Keshmiri, M., Xie, W.-F. "Image-based visual servoing using an optimized trajectory planning technique" *IEEE/ASME Transactions on Mechatronics*, vol. 22(1), pp. 359-370, 2017.

[Kho-2016] Kho, T., Salih, M. H., Woo, Y. S., Ng, Z., Min, J. J., Yee, F. "Enhance implementation of embedded robot auto-navigation system using FPGA for better performance," International Conference on Electronic Design, Phuket, pp. 309-314, 2016.

[Kim-2007] Kim, J. S., Jeon, H. W., Jung, S. "Hardware implementation of nonlinear PID controller with FPGA based on floating point operation for 6-DOF manipulator robot arm". International Conference on Control, Automation and Systems, pp. 1066-1071, 2007.

[Kiran-2013] Kiran, D., Rasheed, A.I., Ramasangu, H. "FPGA implementation of blob detection algorithm for object detection in visual navigation". International

conference on Circuits, Controls and Communications, Channasandra Bengaluru, India, pp. 1–5, 2013.

[Klaiber-2013] Klaiber, M.J., Bailey, D.G., Ahmed, S., Baroud, Y., Simon, S. "A high-throughput FPGA architecture for parallel connected components analysis based on label reuse". International Conference on Field-Programmable Technology, Kyoto, Japan, pp. 302–305, 2013.

[Klaiber-2016] Klaiber, M.J., Bailey, D.G., Baroud, Y.O., Simon, S. "A resource-efficient hardware architecture for connected component analysis" IEEE Transactions on Circuits and Systems for Video Technology, vol. 26(7), pp. 1334-1349, 2016.

[Kofi-2008] Kofi A., Andrew H., Patrick D., Jonathan O. "A run-length based connected component algorithm for FPGA implementation. International Conference on Field-Programmable Technology. Taibei, pp. 177-184, 2008.

[Kofi-2010] Kofi A., Andrew H., Patrick D., Hongying M. "Accelerated hardware video object segmentation: from foreground detection to connected components labelling". *Computer Vision Image Und*. Vol. 114, pp. 1282-1291, 2010.

[Kosmopoulos-2011] Kosmopoulos, D.I. "Robust Jacobian matrix estimation for image-based visual servoing" *Robotics and Computer-Integrated Manufacturing*, vol. 27(1), pp. 82-87, 2011.

[KumarS-2010] Kumar, S., Premkumar, P., Dutta, A., & Behera, L. "Visual motor control of a 7DOF redundant manipulator using redundancy preserving learning network". *Robotica*, vol. 28(06), pp. 795-810, 2010.

[KumarV-2015] Kumar V., Todorov E. MuJoCo "HAPTIX: A virtual reality system for hand manipulation" IEEE-RAS International Conference on Humanoid Robots (Humanoids); Seoul, Korea, pp. 657–663, 2015.

[Kung-2014] Kung, Y.-S., Linh, B.T.H., Wu, M.-K., Lee, F.-C., Chen, W.-C. "FPGA-realization of inverse kinematics control IP for articulated and SCARA robot" *Advanced Structured Materials*, vol. 54, pp. 205-213, 2014.

[Kuon-2007] Kuon, I., Rose, J. "Measuring the gap between FPGAs and ASICs" *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26 (2), pp. 203-215, 2007.

[Kyrkou-2011] Kyrkou C., Theocharides T. "A flexible parallel hardware architecture for AdaBoost-based real-time object detection". *IEEE Trans. Very Large Scale Integr. (VLSI) Systems*, vol. 19, pp. 1034–1047, 2011.

[Lalpuriya-2013] Lalpuriya, K., Prakash, N. R., Rastogi, N. "FPGA based control system for Robotic Applications", *International Journal of Emerging Technology and Advanced Engineering*, vol. 4(6), pp. 2864–2867, 2013.

[Larouche-2015] Larouche, B.P., Zhu, Z.H. "Position-based visual servoing in robotic capture of moving target enhanced by Kalman filter, *International Journal of Robotics and Automation*, vol. 30(3), pp. 267-277, 2015.

[LeeD-2012] Lee, D., Kim, J., Kim, H.J. "Depth estimation for image-based visual servoing of an under-actuated system" *Journal of Institute of Control, Robotics and Systems*, vol. 18(1), pp. 42-46, 2012.

[LeeJ-2013] Lee, J., Lee, J.M. "A study on the visual servoing of autonomous mobile inverted pendulum" *Journal of Institute of Control, Robotics and Systems*, vol. 19(3), pp. 240-247, 2013.

[LeePJ-2012] Lee, P.-J., Lee, M.-S., Wang, R.-C. "A fuzzy control based robotic fish with multiple actuators" *International Journal of Fuzzy Systems*, vol. 14(1), pp. 45-53, 2012.

[LeeWK-2006] Lee, W. K., Jung, S., "FPGA Design for Controlling Humanoid Robot Arms by Exoskeleton Motion Capture System", IEEE International Conference on Robotics and Biomimetics, pp. 1378 – 1383, 2006.

[Lentaris-2012] Lentaris, G., Diamantopoulos, D., Stamoulias, G., Siozios, K., Soudris, D., Rodrigalvarez, M.A. "FPGA-based path-planning of high mobility rover for future planetary missions". IEEE International Conference on Electronics, Circuits, and Systems, Sevilla, Spain, pp. 85-88, 2012.

[Li-2003] Li, T.-H.S., Chang, S.-J., Chen, Y.-X. "Implementation of human-like driving skills by autonomous fuzzy behavior control on an FPGA-based car-like mobile robot" *IEEE Transactions on Industrial Electronics*, vol. 50(5), pp. 867-880, 2003.

[Lippiello-2003] Lippiello, V., Siciliano, B., Villani, L., "Robust Visual Tracking Using a Fixed Multi-Camera System", IEEE International Conference on Robotics and Automation, Taipei, Taiwan, pp. 3333-3338, 2003.

[Lippiello-2007] Lippiello, V., Siciliano, B., Villani, L., "Position-Based Visual Servoing in Industrial Multirobot Cells Using a Hybrid Camera Configuration". IEEE Transactions on Robotics, vol. 23(1) pp. 73-86, 2007.

[Lippiello-2016] Lippiello, V., Cacace, J., Santamaria-Navarro, A., Andrade-Cetto, J., Trujillo, M.A., Esteves, Y.R., Viguria, A. "Hybrid Visual Servoing With Hierarchical Task Composition for Aerial Manipulation" *IEEE Robotics and Automation Letters*, vol. 1(1), pp. 259-266, 2016.

[LiuC-2012] Liu, C., Huang, X., Wang, M. "Target tracking for visual servoing systems based on an adaptive Kalman filter", *International Journal of Advanced Robotic Systems*, vol. 9, 2012.

[LiuH-2007] Liu, H., Meusel, P., Seitz, N., Willberg, B., Hirzinger, G., Jin, M.H., Liu, Y.W., Wei, R., Xie, Z.W. "The modular multisensory DLR-HIT-Hand" *Mechanism and Machine Theory*, vol. 42(5), pp. 612-625, 2007.

[LiuH-2008] Liu, H., Meusel, P., Hirzinger, G., Jin, M., Liu, Y., Xie, Z. "The modular multisensory DLR-HIT-Hand: Hardware and software architecture" *IEEE/ASME Transactions on Mechatronics*, vol. 13(4), pp. 461-469, 2008.

[Liyanage-2014] Liyanage, M. H., Krouglicof, N., "A Single Time Scale Visual Servoing System for a High Speed SCARA Type Robotic Arm" IEEE International Conference on Robotics & Automation, pp. 4153 – 4160, 2014.

[Lopez-Buedo-2004] Lopez-Buedo, S, Boemo, E. "Making Visible the Thermal Behaviour of Embedded Microprocessors on FPGAs. A Progress Report", ACM/SIGDA

International Symposium on Field Programmable Gate Arrays, Monterrey, CA, USA, pp. 79-86, 2004.

[Lopez-Buedo-2002] Lopez-Buedo, S., Garrido, J., Boemo, E.I. "Dynamically inserting, operating, and eliminating thermal sensors of FPGA-based systems". *IEEE Transactions Compon. Packag. Technol,* vol. 25, pp. 561-566, 2002.

[Lorenz-2013] Lorenz, M.G., Mengibar-Pozo, L., Izquierdo-Gil, M.A. "High resolution simultaneous dual liquid level measurement system with CMOS camera and FPGA hardware processor". *Sensors and Actuators A: Physical*, vol. 201, pp. 468-476, 2013.

[Lu-2013] Lu, X., Song, L, Shen, S., He, K., Yu, S., Ling, N. "Parallel Hough Transform-based straight line detection and its FPGA implementation in embedded vision". *Sensors*, vol. 13, pp. 9223-9247, 2013.

[Lucas-1981] Lucas, B. D., Kanade, T. "An iterative image registration technique with an application to stereo vision". Imaging Understanding Workshop, Washington, D.C., USA, pp. 121-130, 1981.

[MaN-2008] Ma, N., Bailey, D. G., Johnston, C. "Optimised single pass connected components analysis". *International Conference on Field Programmable Technology*, pp. 185-192, 2008.

[MaZ-2015] Ma, Z., Su, J. "Robust uncalibrated visual servoing control based on disturbance observer" *ISA Transactions*, vol. 59, pp. 193-204, 2015.

[MacCleery-2008] MacCleery, B., Kassas, Z. M. "New mechatronics development techniques for FPGA-based control and simulation of electromechanical systems", IFAC World Congress, Seoul, Korea, 2008.

[Magdaleno-2010] Magdaleno, E., Lüke, J.P., Rodríguez, M., Rodríguez-Ramos, J.M. "Design of Belief Propagation Based on FPGA for the Multistereo CAFADIS Camera". *Sensors*, vol. 10, pp. 9194-9210, 2010.

[Mahlknecht-2004] Mahlknecht S., Oberhammer R., Novak G. "A real-time image recognition system for tiny autonomous mobile robots" IEEE Real-Time and

Embedded Technology and Applications Symposium, Toronto, ON, Canada. pp. 324–330, 2004.

[Mahony-2012] Mahony, R., Kumar, V., Corke, P. "Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor," *IEEE Robotics & Automation Magazine*, vol. 19(3), pp. 20-32, 2012.

[Mahony-2017] Mahony, R. "Modular design of image based visual servo control for dynamic mechanical systems" Springer Tracts in Advanced Robotics, vol. 100, pp. 129-146, 2017.

[Malis-2003] Malis, E., Rives, P., "Robustness of Image-Based Visual Servoing with Respect to Depth Distribution Errors", IEEE International Conference on Robotics and Automation, Taipei, Taiwan, pp. 1056-1061, 2003.

[Mansard-2005] Mansard, N., Chaumette, F. "Directional redundancy: a new approach of the redundancy formalism". IEEE Conference on Decision and Control, and European Control Conference, pp. 5366-5371, 2005.

[Mansard-2009] Mansard, N., Khatib, O., Kheddar, A., "A Unified Approach to Integrate Unilateral Constraints in the Stack of Tasks". IEEE Transactions on Robotics, vol. 25(3) pp. 670-685, 2009.

[Marchand-1996] Marchand, E., Rizzo, A., Chaumette, F., "Avoiding Robot Joint Limits and Kinematic Singularities in Visual Servoing", 13th International Conference on Pattern Recognition, Vienna, Austria, pp. 297-301, 1996.

[Marín-2007] Marín, R., León, G., Wirz, R., Sales, J., Claver, J.M., Sanz, P.J. "Remote Control within the UJI Robotics Manufacturing Cell using FPGA-based vision". European Control Conference, Kos, Greece, 2007.

[Marín-2009] Marin, R., León, G., Wirz, R., Sales, J., Claver, J.M., Sanz, P.J., Fernández, J. "Remote Programming of Network Robots Within the UJI Industrial Robotics Telelaboratory : FPGA Vision and SNRP Network Protocol". *IEEE Transactions Industrial Electronics.* vol. 56, pp. 4806-4816, 2009

[Mariottini-2008] Mariottini, G. L., Prattichizzo, D., "Image-Based Visual Servoing with Central Catadioptric Cameras". *International Journal of Robotics Research*, vol. 27(1) pp. 41-56, 2008.

[Martin-2010] Martin, Y., Rodríguez-Ramos, L.F., Martín, Y., Díaz, J.J., Piqueras, J., García-Jiménez, J., Rodríguez-Ramos, J.M. "FPGA-based real time processing of the Plenoptic Wavefront Sensor", Southern Programmable Logic Conference (SPL), Ipojuca, Brazil, pp. 87-92, 2010.

[Martinet-2015] Martinet, P. "Comparison of visual servoing techniques: Experimental results" *European Control Conference*, pp. 4555-4560, 2015.

[Masmoudi-2017] Masmoudi M.S., Krichen N., Koesdwiady A.B., Karray F., Masmoudi M. "Design and FPGA Implementation of a Fuzzy-PI Controller for Omnidirectional Robot System". In: Kim JH., Karray F., Jo J., Sincak P., Myung H. (eds) Robot Intelligence Technology and Applications 4. Advances in Intelligent Systems and Computing, vol 447. Springer, Cham, 2017.

[Mauch-2014] Mauch, S., Reger, J. "Real-Time Spot Detection and Ordering for a ShackHartmann Wavefront Sensor with a Low-Cost FPGA". *IEEE Trans. Instrum. Meas.*, vol. 63, pp. 2379–2386, 2014.

[Mezouar-2002] Mezouar, Y., Chaumette, F. "Avoiding self-occlusions and preserving visibility by path planning in the image". *Robotics and Autonomous Systems*, vol. 41(2), pp. 77-87, 2002.

[Mezouar-2003] Mezouar, Y., Chaumette, F., "Optimal Camera Trajectory with Image-Based Control". *International Journal of Robotics Research*, vol. 22(10-11) pp. 781-803, 2003.

[Miljković-2013] Miljković, Z., Vuković, N., Mitić, M., Babić, B. "New hybrid vision-based control approach for automated guided vehicles" *International Journal of Advanced Manufacturing Technology*, vol. 66 (1-4), pp. 231-249, 2013.

[Mitsubishi-2005] Mitsubishi Heavy Industries, LTD. General Purpose Robot PA10 Series. Instruction manual for servo driver. SKC-GC20004, REV. 0, 2005.

[Miyazaki-1990] Miyazaki, F., Masutani, Y., *"Robustness of Sensory Feedback Control Based on Imperfect Jacobian"*, international symposium on Robotics research, pp. 201-208, 1990.

[Mohta-2014] Mohta, K., Kumar, V., Daniilidis, K. "Vision-based control of a quadrotor for perching on lines" IEEE International Conference on Robotics and Automation, pp. 3130-3136, 2014.

[Mommasson-2007] Monmasson, E., Cirstea, M.N. "FPGA Design Methodology for Industrial Control Systems-A Review". *IEEE Transactions on Industrial Electronics* vol. 54, pp. 1824-1842, 2007.

[Morales-Velázquez-2010] Morales-Velázquez, L., de Jesus Romero-Troncoso, R., Osornio-Rios, R. A., Herrera-Ruiz, G., Cabal-Yepez, E., "Open-architecture system based on a reconfigurable hardware–software multi-agent platform for CNC machines". *Journal of Systems Architecture*, vol. 56, pp. 407–418, 2010.

[Moreno-2010] Moreno, S.V., Vera, L.A., Osornio, R.A., Dominguez, A., Stiharu, I., Romero, R. "A field programmable gate array-based reconfigurable smart-sensor network for wireless monitoring of new generation computer numerically controlled machines". *Sensors*, vol. 10, pp. 7263-7286, 2010.

[Mukay-2006] Mukai, S., Maru, N. "Redundant arm control by linear visual servoing using pseudo inverse matrix". IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1237-1242, 2006.

[Musić-2014] Musić, J., Bonković, M., Cecić, M. "Comparison of uncalibrated model-free visual servoing methods for small-amplitude movements: A simulation study" *International Journal of Advanced Robotic Systems*, vol. 11(1), 2014.

[Muthuramalingam-2008] Muthuramalingam, A., Himavathi, S., Srinivasan, E. "Neural Network Implementation Using FPGA: Issues and Application". *International Journal of Information Technology*, vol. 4, pp. 86-92, 2008.

[Nadi-2014] Nadi, F., Derhami, V., Rezaeian, M. "Visual servoing control of robot manipulator with Jacobian matrix estimation" RSI/ISM International Conference on Robotics and Mechatronics, pp. 405-409, 2014.

[Nematollahi-2009] Nematollahi, E., Janabi-Sharifi, F., "Generalizations to Control Laws of Image-Based Visual Servoing". *International Journal of Optomechatronics*, vol. 3(3) pp. 167-186, 2009.

[NguyenKQ-2014] Nguyen, K. Q., Nguyen, T. H., Ha, Q. P., "FPGA-Based Sensorless PMSM Speed Control Using Reduced-Order Extended Kalman Filters" *IEEE Transactions on Industrial Electronics*, vol.61(12), pp.6574-6582, 2014.

[NguyenTB-2009] Nguyen, T.B., Chung, S.T. "An Improved Real-Time Blob Detection for Visual Surveillance". International Congress on Image and Signal Processing, Tianjin, China, pp. 1–5, 2009.

[Oberkampf-1996] Oberkampf, D., DeMenthon, D. F., Davis, L. S., "Iterative Pose Estimation Using Coplanar Feature Points". *Computer Vision and Image Understanding*, vol. 63(3) pp. 495-511, 1996.

[Oro-2011] Oro, D., Fernandez, C., Saeta, J.R., Martorell, X., Hernando, J. "Real-time GPU-based face detection in HD video sequences". IEEE International Conference on Computer Vision Workshops, Barcelona, Spain, pp. 530–537, 2011.

[Ortigosa-2011] Ortigosa E.M., Cañas, A., Ros, E., Carrillo, R.R. "FPGA Implementation of a Perceptron-Like Neural Network for Embedded Applications". In *Artificial Neural Nets Problem Solving Methods*; Mira, J., Álvarez, J.R., Eds.; Springer Berlin Heidelberg: Berlin, Germany, vol. 2687, pp. 1-8, 2011.

[Osornio-Rios-2007] Osornio-Rios, R.A., de Jesus Romero-Troncoso, R., Herrera-Ruiz, G., Castañeda-Miranda, R. "Computationally efficient parametric analysis of discrete-time polynomial based acceleration-deceleration profile generation for industrial robotics and CNC machinery" *Mechatronics*, vol. 17(9), pp. 511-523, 2017.

[Osornio-Rios-2009] Osornio-Rios, R.A., de Jesús Romero-Troncoso, R., Herrera-Ruiz, G., Castañeda-Miranda, R. "FPGA implementation of higher degree polynomial

acceleration profiles for peak jerk reduction in servomotors" *Robotics and Computer-Integrated Manufacturing*, vol. 25(2), pp. 379-392, 2009.

[Osuna-2012] Osuna, C.G., Marcos, M.S., Ituero, P., Lopez-Vallejo, M. "A monitoring infrastructure for FPGA self-awareness and dynamic adaptation". IEEE International Conference on Electronics, Circuits, and Systems, Sevilla, Spain, pp. 765-768, 2012.

[Ozawa-2012] Ozawa, R., Chaumette, "F. Dynamic visual servoing with image moments for a quadrotor using a virtual spring approach" IEEE International Conference on Robotics and Automation, pp. 5670-5676, 2011.

[Palmieri-2012] Palmieri, G., Palpacelli, M., Battistelli, M., Callegari, M. "A comparison between position-based and image-based dynamic visual servoings in the control of a translating parallel manipulator" *Journal of Robotics*, 2012.

[Paralic-2012] Paralic, M. "Fast connected component labeling in binary images". International Conference on Telecommunications and Signal Processing (TSP), Prague, Czech Republic, pp. 706–709, 2012.

[Parsapour-2013] Parsapour, M., RayatDoost, S., Taghirad, H.D. "Position based sliding mode control for visual servoing system" International Conference on Robotics and Mechatronics, , pp. 337-342, 2013.

[ParkJH-2009] Park, J.H., Lee, Y.J. "Robust visual servoing for motion control of the ball on a plate," *Mechatronics*, vol. 13(7), pp. 723–738, 2009.

[ParkDH-2012] Park, D. H., Kwon, J. H., Ha, I. J. "Novel Position-Based Visual Servoing Approach to Robust Global Stability Under Field-of-View Constraint," *IEEE Transactions on Industrial Electronics*, vol. 59(12), pp. 4735-4752, 2012.

[Patro-2014] Patro B.N. "Design and implementation of novel image segmentation and BLOB detection algorithm for real-time video surveillance using DaVinci processor"; International Conference on Advances in Computing, Communications and Informatics, Delhi, India, pp. 1909–1915, 2014.

[Peng-2016] Peng, W., Jian, W., Yuan, Z., Jixiang, L., Peng, Z. "FPGA implementation of image acquisition for quadruped search robot monitor" International Journal of Smart Home, vol. 10(3), pp. 159-170, 2016.

[PérezC-2008] Pérez, C. "Control de robots manipuladores usando información visual: Aplicación a la estimación del movimiento del objeto". Tesis Doctoral, 2008.

[PérezJ-2015a] Pérez, J., Alabdo, A., Garcia, G.J., Pomares, J., Torres, F. "FPGA-based visual control of robot manipulators using dynamic perceptibility" International Conference on ReConFigurable Computing and FPGAs, 2015.

[PérezJ-2015b] Pérez, J., Alabdo, A., Garcia, G.J., Pomares, J., Torres, "ViSeC-Matlab: Una herramienta para el aprendizaje de sistemas de con-trol visual sobre Matlab", XXXVI Jornadas de Automática, Bilbao, Spain, 2015.

[PérezJ-2016] Pérez, J., Alabdo, A., Pomares, J., García, G.J., Torres, F. "FPGA-based visual control system using dynamic perceptibility" *Robotics and Computer-Integrated Manufacturing*, vol. 41, pp. 13-22, 2016.

[PérezJM-2009b] Pérez, J.M., Sánchez, P., Martínez, M. "High-Definition Belief-Propagation based Stereo Matching FPGA architecture". Conference on Design of Circuits and Integrated System, Zaragoza, Spain, 2009.

[Pérez-Peña-2013] Perez-Peña, F., Morgado-Estevez, A., Linares-Barranco, A., Jiménez-Fernández, A., Lopez-Coronado, J., Muñoz-Lozano, J.L. "A FPGA spike-based robot controlled with neuro-inspired VITE". In *Advances in Computational Intelligence*; Rojas, I., Joya, G., Gabestany, J., Eds.; Springer Berlin Heidelberg: Berlin, Germany, vol. 7902, pp. 299-308, 2013.

[Photonfocus-2004] Photonfocus. MV-D752-160 User's Manual. 2004.

[Pierce-2014] Pierce, B., Cheng, G. "Versatile modular electronics for rapid design and development of humanoid robotic subsystems". IEEE/ASME International Conference on Advanced Intelligent Mechatronics, pp. 735-741, 2014.

[Pomares-2011a] Pomares, J., Corrales, J. A., García, G. J., Torres, F. "Direct visual Servoing to track trajectories in human-robot cooperation". *International Journal of Advanced Robotic Systems*, vol. 8(4), pp. 129-138, 2011.

[Pomares-2014] Pomares, J., Perea, I., Torres, F. "Dynamic Visual Servoing With Chaos Control for Redundant Robots," *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 2, pp 423-431, 2014.

[Prats-2008] Prats, M., Martinet, P., Del Pobil, A. P., Lee, S., "Robotic Execution of Everyday Tasks by Means of External Vision/Force Control". *Intelligent Service Robotics*, vol. 1(3) pp. 253-266, 2008.

[Pritschow-1990] Pritschow G. "Automation technology—on the way to an open system architecture". *Robotics and Computer Integrated Manufacturing*, vol. 7(1/ 2), pp.103–111, 1990.

[Pujari-2014] Pujari, S., Nayak, S. "Implementation of edge detection using FPGA & model based approach". International Conference on Information Communication and Embedded Systems, Chennai, India, 2014.

[Pullteap-2016] Pullteap, S. "Development of a walking robot by using FPGA controller". IEEE International Conference on Research and Education in Mechatronics, pp. 054-057, 2016.

[Pyragas-2006] Pyragas, K. "Delayed feedback control of chaos," *Philosophical Transactions of the Royal Society*, vol. 364(1846), pp. 2309-2334, 2006.

[Reaz-2005] Reaz, M.B.I., Mohd-Yasin F., Tan, S.L., Tan, H.Y., Ibrahimy, M.I. "Partial Encryption of Compressed Images Employing FPGA". IEEE International Symposium on Circuits and Systems, Kobe, Japan, pp. 2385-2388, 2005.

[Reza-2004] Reza, A. M. "Realization of the Contrast Limited Adaptive Histogram Equalization (CLAHE) for real-time image enhancement". *Journal of VLSI Signal Processing Systems for Signal Image and Video Technology*, vol. 38, pp. 35-44, 2004.

[Ribo-2004] Ribo, M., Brandner, M., Pinz, A. "A flexible software architecture for hybrid tracking". *Journal of Field Robotics*, vol. 21(2), pp. 53-62, 2004.

[Rodríguez-2008] Rodríguez, J.M., Castelló, E.M., Conde, C.D., Valido, M.R., Marichal-Hernández, J.G. "2D-FFT implementation on FPGA for wavefront phase recovery from the CAFADIS camera". SPIE 7015, Adaptive Optics Systems, 701539, Marseille, France, 2008.

[Rodriguez-Andina-2007] Rodriguez-Andina, J.J., Moure, M.J., Valdes, M.D. "Features, design tools, and application domains of FPGAs" *IEEE Transactions on Industrial Electronics*, vol. 54(4), pp. 1810-1823, 2007.

[Rodríguez-Ramos-2006] Rodríguez-Ramos, L.F., Viera, T., Herrera, G., Gigante, J.V., Gago, F., Alonso, A. "Testing FPGAs for real-time control of adaptive optics in giant telescopes". SPIE 6272, Adaptive Optics II, 62723X, Orlando, FL, USA, 2006.

[Rodríguez-Ramos-2008] Rodríguez-Ramos, L.F., Díaz, J.J., Piqueras, J.J., Martin, Y., Rodríguez-Ramos, J.M. "FPGA–based slope computation for ELTs adaptive optics wavefront sensors". SPIE 7015, Adaptive Optics Systems, 701530, Marseille, France, 2008.

[Roetenberg-2007] Roetenberg, D., Slycke, P. J., Veltink, P. H. "Ambulatory position and orientation tracking fusing magnetic and inertial sensing". *IEEE Transactions on Biomedical Engineering*, vol. 54(5), pp. 883-890, 2007.

[Rosado-2012] Rosado, A., Bataller, M., Guerrero, J. "FPGA implementation of Spiking Neural Network". IFAC conference on Embedded Systems, Computational Intelligence and Telematics in Control, Wurzburg, Germany, pp. 139-144, 2012.

[Roshandel-2015] Roshandel, N. T., Dinavahi, V. "A General Framework for FPGA-Based Real-Time Emulation of Electrical Machines for HIL Applications," *IEEE Transactions on Industrial Electronics*, vol. 62(4), pp.2041-2053, 2015.

[Rostro-Gonzalez-2015] Rostro-Gonzalez, H., Cerna-Garcia, P.A., Trejo-Caballero, G., Garcia-Capulin, C.H., Ibarra-Manzano, M.A., Avina-Cervantes, J.G., Torres-Huitzil, C. "A CPG system based on spiking neurons for hexapod robot locomotion" *Neurocomputing*, vol. 170, pp. 47-54, 2015.

[Sabatini-2010] Sabatini, S.P., Gastaldi, G., Solari, F., Pauwels, K., Hulle, M.M.V., Diaz, J., Ros, E., Pugeault, N., Krger, N. "A compact harmonic code for early vision based on anisotropic frequency channels". *Computer Vision Image Understanding*, vol. 114, pp. 681-699, 2010.

[Saeed-2015] Saeed A., Al-Hamadi A., Ghoneim A. "Head Pose Estimation on Top of Haar-Like Face Detection: A Study Using the Kinect Sensor". *Sensors*, vol. 15, pp. 20945–20966, 2015.

[San Woo-2016] San Woo, Y., Salih, M. H., Kho, T., Yee, F., Min, J. J., Ng, Z. "Enhance implementation of flying robot auto-navigation system on FPGA for better performance". IEEE International Conference on Electronic Design, pp. 315-320, 2016.

[SánchezDF-2009] Sánchez, D. F., Muñoz, D. M., Llanos, C. H., Motta, J. M. "FPGA implementation for direct kinematics of a spherical robot manipulator". IEEE International Conference on Reconfigurable Computing and FPGAs, pp. 416-421, 2009.

[SánchezJ-2012] Sánchez, J., Benet, G., Simó, J.E. "Video Sensor Architecture for Surveillance Applications". *Sensors*, vol. 12, pp. 1509-1528, 2012.

[SánchezLAP-2015] Sánchez, L. A. P., Moreno, C. A. A., Daza, H. A. B. "Design and construction of a line follower robot guided by pixels values of a camera connected to an FPGA". Symposium on Signal Processing, Images and Computer Vision, pp. 1-5, 2015.

[SánchezPM-2013] Sanchez, P.M., Machado, O., Bueno-Peña, E.J., Rodríguez, F.J., Meca, F.J. "FPGA-Based Implementation of a Predictive Current Controller for Power Converters". *IEEE Transactions on Industrial Informatics*, vol. 9, pp. 1312-1321, 2013.

[Sánchez-Solano-2007] Sánchez-Solano, S., Cabrera, A.J., Baturone, I., Moreno-Velo, F.J., Brox, M. "FPGA implementation of embedded fuzzy controllers for robotic applications" *IEEE Transactions on Industrial Electronics*, vol. 54(4), pp. 1937-1945, 2007.

[Sánchez-Solano-2013] Sánchez-Solano, S., Brox, M., Toro, E., Brox, P. Baturone, I. "Model-Based Design Methodology for Rapid Development of Fuzzy Controllers on FPGAs". *IEEE Transactions on Industrial Informatics*, vol. 9, pp. 1361-1370, 2013.

[Sawo-2005] Sawo, F., Fujita, M., Sawodny, O. "Passivity-based dynamic visual feedback control of manipulators with kinematic redundancy". IEEE Conference on Control Applications, pp. 1200-1205, 2005.

[Schramm-2006] Schramm, F., Morel, G., "Ensuring Visibility in Calibration-Free Path Planning for Image-Based Visual Servoing", *IEEE Transactions on Robotics*, vol. 22(4) pp. 848-854, 2006.

[Serra-2016] Serra, P., Cunha, R., Hamel, T., Cabecinhas, D., Silvestre, C. "Landing of a quadrotor on a moving target using dynamic image-based visual servo control" *IEEE Transactions on Robotics*, vol. 32(6), pp. 1524-1535, 2016.

[Serrano-Gotarredona-2009] Serrano-Gotarredona, R., Oster, M., Lichtsteiner, P., Linares-Barranco, A., Paz-Vicente, R., Gomez-Rodriguez, F., Linares-Barranco, B. "CAVIAR: a 45k neuron, 5M synapse, 12G connects/s AER hardware sensory-processing- learning-actuating system for high-speed visual object recognition and tracking". *IEEE transactions on neural networks,* vol. 20, pp. 1417-1438, 2009.

[Shademan-2012] Shademan, A., Jagersand, M. "Robust sampling-based planning for uncalibrated visual servoing", IEEE International Conference on Intelligent Robots and Systems, pp. 2663-2669, 2012.

[Shao-2006] Shao, X., Sun, D., Mills, J. K., "A New Motion Control Hardware Architecture with FPGA-Based IC Design for Robotic Manipulators". *International Conference on Robotics and Automation*, pp. 3520-3525, 2006.

[Shao-2007] Shao, X., Sun, D. "Development of a new robot controller architecture with FPGA-based IC design for improved high-speed performance", *IEEE Transactions on Industrial Informatics*, vol. 3(4), pp. 312-321, 2007.

[Siciliano-2010] Siciliano B., Villani L., Lippiello V., De Santis A. "Force and Visual Control for Safe Human-Robot Interaction". En: Angeles J., Boulet B., Clark J.J.,

Kövecses J., Siddiqi K. (eds) Brain, Body and Machine. Advances in Intelligent and Soft Computing, vol 83. Springer, Berlin, Heidelberg, 2010.

[Simon-2001] Simon, D. "Kalman filtering. Embedded systems programming", vol. 14(6), pp. 72-79, 2001.

[Siradjuddin-2012] Siradjuddin, I., McGinnity, T.M., Coleman, S., Behera, L. "An iterative robot-image Jacobian approximation of image-based visual servoing for joint limit avoidance" *International Journal of Mechatronics and Automation*, vol. 2(4), pp. 227-239, 2012.

[Siradjuddin-2014] Siradjuddin, I., Behera, L., McGinnity, T.M., Coleman, S. "Image-based visual servoing of a 7-DOF robot manipulator using an adaptive distributed fuzzy PD controller" *IEEE/ASME Transactions on Mechatronics*, vol. 19(2), pp. 512-523, 2014.

[Spong-1989] Spong, M. W., "Robot Dynamics and Control". John Wiley & Sons, Inc.: New York, NY, USA, pp. 336, 1989.

[Sudha-2015] Sudha, N., Santhiyakumari, N., Lay, B. "Segmentation of bowel images and its implementation using Virtex FPGA kit". IEEE International Conference on Electrical, Computer and Communication Technologies, Coimbatore, India, 2015.

[Sulaiman-2009] Sulaiman, N., Obaid, Z. A., Marhaban, M. H., Hamidon, M. N. "FPGA-Based Fuzzy Logic: Design and Applications - a Review". *International Journal of Engineering and Technology*, vol. 1, pp. 491-503, 2009.

[Swati-2014] Swati, Dixit, G. "Improved algorithm for blob detection in document images". International Conference Confluence The Next Generation Information Technology Summit, Noida, India, pp. 703–708, 2014.

[Tahri-2009] Tahri, O., Mezouar, Y., Andreff, N., Martinet, P., "Omnidirectional Visual-Servo of a Gough-Stewart Platform". *IEEE Transactions on Robotics*, vol. 25(1) pp. 178-183, 2009.

[Tahri-2012] Tahri, O., Araujo, H. "Non-central catadioptric cameras visual servoing for mobile robots using a radial camera model", IEEE International Conference on Intelligent Robots and Systems, pp. 1683-1688, 2012.

[Tahri-2013] Tahri, O., Araujo, H., Chaumette, F., Mezouar, Y. "Robust image-based visual servoing using invariant visual information", *Robotics and Autonomous Systems*, vol. 61(12), pp. 1588-1600, 2013.

[Takegaki-1981] Takegaki, M., Arimoto, S., "A New Feedback Method for Dynamic Control of Manipulators". *Journal of Dynamic Systems, Measurement, and Control*, vol. 103(2) pp. 119-125, 1981.

[Tao-2016] Tao, B., Gong, Z., Ding, H. "Survey on uncalibrated robot visual servoing control" *Lixue Xuebao/Chinese Journal of Theoretical and Applied Mechanics*, vol. 48(4), pp. 767-783, 2016.

[Tapia-Espinoza-2013] Tapia-Espinoza R., Torres-Torriti M. "Robust Lane Sensing and Departure Warning under Shadows and Occlusions". *Sensors*. vol. 13, pp. 3270–3298, 2013.

[Thuilot-2002] Thuilot, B., Martinet, P., Cordesses, L., Gallice, J., "Position Based Visual Servoing: Keeping the Object in the Field of Vision", IEEE International Conference on Robotics and Automation, Washington, DC, USA, pp. 1624-1629, 2002.

[Tong-2006] Tong, J.G., Anderson, I.D.L., Khalid, M.A.S. "Soft-core processors for embedded systems", Proceedings of the International Conference on Microelectronics, pp. 170-173, 2006.

[Torres-2014] Torres, A., Alacid, B., Alabdo, A., García, G.J., "Tracking de puntos para un sistema de control visual embebido en FPGA", Semana de la Automática y la Robótica de Alicante, Alicante, España, pp. 81-85, 2014.

[Trein-2008] JTrein, Schwarzbacher, A. T., Hoppe, B. "FPGA Implementation of a Single Pass Real-Time Blob Analysis Using Run Length Encoding", MPC-Workshop, Ravensburg-Weingarten, Germany, 2008.

[Trujano-2011] Trujano, M. A., Garrido, R., Soria, A. "Visual PID Control of a Redundant Planar Parallel Robot". ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, pp. 589-598, 2011.

[Tu-2011] Tu, Y. W., Ho, M. T. "Design and implementation of robust visual servoing control of an inverted pendulum with an FPGA-based image co-processor". *Mechatronics*, vol. 21, pp. 1170-1182, 2011.

[Udwadia-2003] Udwadia, F. E. "A new perspective on tracking control of nonlinear structural and mechanical systems", *Proc. of the Royal Society of London Series* A, pp. 1783-1800, 2003.

[Vachhani-2009] Vachhani, L., Sridharan, K., Meher, P. K., "Efficient FPGA Realization of CORDIC With Application to Robotic Exploration," *IEEE Transactions on Industrial Electronics*, vol.56(12), pp.4915-4929, 2009.

[Van-2016] Van, M., Wu, D., Ge, S.S., Ren, H. "Fault Diagnosis in Image-Based Visual Servoing with Eye-in-Hand Configurations Using Kalman Filter", *IEEE Transactions on Industrial Informatics*, 12 (6), pp. 1998-2007, 2016.

[Vicente-2009] Vicente, A.G., Muñoz, I.B., Molina, P.J., Lázaro-Galilea, J.L. "Embedded Vision Modules for Tracking and Counting People". *IEEE Transaction on Instrumentation and Measurement*, vol. 58, pp. 3004-3011, 2009.

[Viola-2004] Viola P., Jones M. "Robust real-time face detection". *International Journal Compututer Vision*, vol. 57, pp.137–154, 2004.

[Volder-1959] Volder, J.E. "The CORDIC Trigonometric Computing Technique" *IRE Transactions on Electronic Computers*, vol. EC-8 (3), pp. 330-334, 1959.

[WangH-2014] Wang, H., Chen, W., Wang, C., Wang, X., Pfeifer, R. "Dynamic modeling and image-based adaptive visual servoing of cable-driven soft robotic manipulator" *IFAC Proceedings Volumes*, vol. 19, pp. 11884-11889, 2014.

[WangK-2014] Wang, K., Liu, Y., Li, L. "Visual servoing trajectory tracking of nonholonomic mobile robots without direct position measurement" *IEEE Transactions on Robotics*, vol. 30 (4), pp. 1026-1035, 2014.

[WangS-2016] Wang, S., Ye, A., Guo, H., Gu, J., Wang, X., Yuan, K. "Autonomous pallet localization and picking for industrial forklifts based on the line structured light," IEEE International Conference on Mechatronics and Automation, Harbin, pp. 707-713, 2016.

[WangY-2009] Wang, Y., Lang, H. X., de Silva, C. W., "A Robust Mobile Robot Manipulation System Using a Switch-Based Visual-Servo Controller", International Conference on Industrial Mechatronics and Automation, Chengdu, Peoples R China, pp. 364-367, 2009.

[WangY-2012] Wang, Y., Thunberg, J., Hu, X. "A transformation of the Position Based Visual Servoing Problem into a convex optimization problem," IEEE Conference on Decision and Control (CDC), Maui, HI, pp. 5673-5678, 2012.

[WangY-2013] Wang, Y., Liu, L., Yin, S., Zhu, M., Cao, P., Yang, J., Wei, S. "The organization of on-chip data memory in one coarse-grained reconfigurable architecture" *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E96-A (11), pp. 2218-2229, 2013.

[WangZ-2015] Wang, Z. "Hardware implementation for a hand recognition system on FPGA". International Conference on Electronics Information and Emergency Communication, Beijing, China, 2015.

[Wei-2015] Wei, L., Gregoire, L. A., Belanger, J. "A Modular Multilevel Converter Pulse Generation and Capacitor Voltage Balance Method Optimized for FPGA Implementation," *IEEE Transactions on Industrial Electronics*, vol. 62(5), pp.2859-2867, 2015.

[Wen-Hong-2013] Wen-Hong, Z., Lamarche, T., Dupuis, E., Jameux, D., Barnard, P., Liu. G. "Precision Control of Modular Robot Manipulators: The VDC Approach With Embedded FPGA". *IEEE Transactions on Robotics*, vol. 29(5), 2013.

[Wilson-1996] Wilson, W. J., Williams Hulls, C. C., Bell, G. S., "Relative End-Effector Control Using Cartesian Position Based Visual Servoing". IEEE Transactions on Robotics and Automation, vol. 12(5) pp. 684-696, 1996.

[Winter-1999] Winter, M. E. "N-FINDR: an algorithm for fast autonomous spectral end-member determination in hyperspectral data". Proceedings of the SPIE 3753, *Imaging Spectrometry V*, 266, Denver, CO, USA, 1999.

[XieWF-2009] Xie, W. F., Li, Z., Tu, X. W., Perron, C., "Switching Control of Image-Based Visual Servoing with Laser Pointer in Robotic Manufacturing Systems". IEEE Transactions on Industrial Electronics, vol. 56(2) pp. 520-529, 2009.

[XieH-2016a] Xie, H., Lynch, A.F., Jagersand, M. "Dynamic IBVS of a rotary wing UAV using line features", Robotica, vol. 34(9), pp. 2009–2026, 2016.

[XieH-2016b] Xie, H., Lynch, A. "Dynamic image-based visual servoing for unmanned aerial vehicles with bounded inputs" Canadian Conference on Electrical and Computer Engineering, 2016.

[Xilinx-2004] Xilinx, Microblaze Processor Reference Guide, 2004.

[Yanga-2012] Yanga, N., Li, D., Zhanga, J., Xi, Y. "Model predictive controller design and implementation on FPGA with application to motor servo system". *Control Engineering Practice*, vol. 20, pp. 1229-1235, 2012.

[Yuhai-2011] Yuhai, L., Mei, K., Dong, P. "An Effiient and Low Memory Requirement Algorithm for Extracting Image Component Information". *International Journal Advanced Intelligence*, vol. 3, pp. 255–267, 2011.

[Zabih-1994] Zabih, R.; Woodfill, J. "Non-parametric Local Transforms for computing Visual Correspondence". European Conference on Computer Vision, Stockholm, Sweden, pp. 150-158, 1994.

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

[ZhangL-2014] Zhang, L., Slaets, P., Bruyninckx, H. "An open embedded industrial robot hardware and software architecture applied to position control and visual servoing application", *International Journal of Mechatronics and Automation*, vol. 4(1), pp.63-72, 2014

[ZhangT-2016] Zhang, T., Jiang, L., Fan, S., Wu, X., Feng, W. "Development and experimental evaluation of multi-fingered robot hand with adaptive impedance control for unknown environment grasping", Robotica, vol. 34(5), pp. 1168-1185, 2016.

[ZhangX-2015] Zhang, X., Fang, Y., Sun, N. "Visual servoing of mobile robots for posture stabilization: From theory to experiments" *International Journal of Robust and Nonlinear Control*, vol. 25(1), pp. 1-15, 2015.

[ZhangX-2016] Zhang, X., Fang, Y., Liang, X., Zhang, X. "Geometric adaptive dynamic visual servoing of a quadrotor UAV", IEEE/ASME International Conference on Advanced Intelligent Mechatronics, pp. 312-317, 2016.

[ZhangY-2014] Zhang Y., Xu X., Lu H., Dai Y. "Two-Stage Obstacle Detection Based on Stereo Vision in Unstructured Environment" International Conference on Intelligent Human-Machine Systems and Cybernetics, Hangzhou, China, pp. 168–172, 2014.

[ZhaoF-2013] Zhao, F., Zhang Lu, H., Zhang, Z. Y. "Real-Time single-pass connected components analysis algorithm" *EURASIP Journal Image and Video Processing*, vol. 2013(21), pp. 1-10, 2013.

[ZhaoYM-2017] Zhao, Y.M., Lin, Y., Xi, F., Guo, S., Ouyang, P. "Switch-Based Sliding Mode Control for Position-Based Visual Servoing of Robotic Riveting System" *Journal of Manufacturing Science and Engineering*, vol. 139 (4), 2017.

[Zhu-2003] Zhu, J., Sutton, P. "FPGA Implementations of Neural Networks - A Survey of a Decade of Progress". *Field Programmable Logic and Application, Lecture Notes in Computer Science*; Cheung, P.Y.K., Constantinides, G.A., de Sousa, J.T., Eds.; Springer: Lisboa, Portugal, vol. 2778, pp. 1062-1066, 2003.

[Zouari-2014] Zouari, L., Ayed, M. B., Abid, M. " Embedded control of robot arm driven by Brushless DC motor on FPGA". World Conference Complex Systems, pp. 722-727, 2014.

Universitat d'Alacant
Universidad de Alicante

Escola de Doctorat
Escuela de Doctorado

ED|UA

edua.ua.es