https://doi.org/10.17979/spudc.9788497498081.1037

CONTROL VISUAL DINÁMICO BASADO EN FPGA DE UN ROBOT MANIPULADOR DE 6 GRADOS DE LIBERTAD

Aiman Alabdo, Javier Pérez, Jorge Pomares, Gabriel J. García, Fernando Torres
Departamento de Física, Ingeniería de Sistemas y Teoría de la Señal
Carretera San Vicente del Raspeig, s/n, San Vicente del Raspeig, CP: 03690
Universidad de Alicante
{aiman.alabdo, jpalepuz, jpomares, gjgg, Fernando.Torres}@ua.es

Resumen

En este artículo se describe la formulación, implementación y experimentación de un sistema de control visual dinámico aplicado a un robot de 6 grados de libertad. Se propone una arquitectura hardware basada en FPGAs para la implementación de los controladores. Con el objetivo de limitar la latencia del controlador se ha implementado en la FPGA no sólo el controlador sino también la captura y procesamiento de la información visual. Se hace uso de las capacidades de procesamiento paralelo de la FPGA para optimizar los diferentes componentes del sistema de control visual propuesto. Finaliza el artículo con los resultados experimentales obtenidos en tareas de posicionamiento de un robot Mitsubishi PA10 de 6 grados de libertad.

Palabras Clave: Control visual, robot manipulador, sistema embebido, FPGA.

1 INTRODUCCIÓN

El uso de Field Programmable Gate Arrays (FPGAs) permite aplicar una tecnología basada en hardware reprogramable específica a la implementación de controladores visuales [1]. A diferencia de los controladores visuales clásicos, la estrategia de control visual directo [2] desempeña el control articular directamente usando información visual, generando los pares a aplicar a las articulaciones. Como se describe a lo largo del artículo, la implementación de un sistema de control visual directo totalmente integrado en una FPGA permite mejorar dichos sistemas reduciendo los retrasos por procesamiento y obteniendo retrasos estables en la retroalimentación. Además, su parcial reconfiguración permite modificar secciones de la lógica implementada en la FPGA, lo que permite ajustar la funcionabilidad y dinámica de los controladores a los requisitos de la aplicación a desarrollar.

Este artículo presenta una arquitectura embebida reconfigurable basada en FPGAs para el control visual directo de robots industriales. Esta arquitectura es rápidamente adaptable para el control de cualquier robot manipulador multi-articular. De esta manera, el sistema es modular y flexible ante posibles cambios que puedan ocurrir como consecuencia de la incorporación o modificación de los drivers de control, o incluso a cambios en la configuración de los sistemas de adquisición de datos y su control. Además, cuando se desarrollan sistemas de control en una FPGA, se debe llevar a cabo un compromiso entre el rendimiento del control y la complejidad de la arquitectura hardware. Para evaluar la arquitectura, dos controladores diferentes han sido implementados. El primero consiste en el bien conocido controlador basado en Jacobiana transpuesta [2] y el segundo que es propuesto en este artículo, el cual desarrolla un framework óptimo. En trabajos previos como [3] este framework es definido para guiar visualmente sistemas robóticos durante la manipulación de objetos rígidos. En este artículo, el framework es modificado para obtener un framework general para brazos robóticos e implementado en una arquitectura basada en FPGA. De este framework se pueden obtener controladores previos ampliamente conocidos. Adicionalmente, este framework también puede ser empleado para definir nuevos controladores con diferentes propiedades dinámicas.

Las FPGAs han sido utilizadas con éxito en diferentes sistemas como el control de velocidad de motores asíncronos de imanes permanentes [4], emulación de máquinas eléctricas [5], implementación de métodos de control para generación de pulsos y balanceo de voltaje [6], etc. Dentro del campo de la robótica de control, es interesante mencionar trabajos previos como [7], 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 integren FPGAs en algunos de sus componentes de su diseño [8]. En este último caso, los algoritmos son particionados en porciones lineales, que son implementados en una FPGA, y en porciones no lineales, las cuales son implementadas en un DSP. El trabajo presentado en [9] especifica las plataformas hardware y software de sistemas de control para robots de alta velocidad usando DSPs. En [10] se describe

una arquitectura de control de robots, la cual puede ser fácilmente aplicada a cualquier robot, cuyos controladores son implementados con una FPGA. Un controlador basado en FPGA para un brazo robótico humanoide que implementa varias tareas como controladores PID, comunicaciones, contadores de encoders o generadores de pulsos PWM son descritos en [11]. En [12] se propone un controlador embebido basado en FPGA para ser aplicado a robots modulares. Sin embargo, a pesar de que los sistemas de control están más integrados en una gran cantidad de aplicaciones, no existen previas experiencias de arquitecturas para el control visual directo de manipuladores robóticos basado en FPGAs. Dentro del ámbito de sistemas de control visual hay sólo unas pocas implementaciones que integran FPGAs en algún componente del sistema de control visual. La mayoría de retrasos en estos sistemas se deben a las tareas de procesamiento de imagen. Es por esto que existen algunos trabajos como [13] que optimiza el proceso de captura de imagen gracias a la implementación hardware permitida por una FPGA. Adicionalmente, es posible encontrar sistemas de control visual indirecto embebidos que integran el procesamiento de imagen, así como el control [14]. Sin embargo, no es posible encontrar arquitecturas para el control visual directo de robots manipuladores.

El resto del artículo se organiza como se indica a continuación: en la Sección 2 se describen los diferentes componentes que integran la arquitectura hardware y software, detallando además los principales componentes del sistema de visión. El sistema de control propuesto se detalla en la Sección 3. La Sección 4 describe diferentes experimentos que demuestran el correcto comportamiento de la arquitectura, así como las diferentes técnicas de optimización desarrolladas. Finalmente, la Sección 5 indica las principales conclusiones del trabajo propuesto en el artículo.

2 ARQUITECTURA DE CONTROL

El sistema de control visual dinámico propuesto se ha embebido en una FPGA. Como se muestra en la Figura 1, el diseño del sistema embebido se compone de 3 partes principales: una FPGA, adaptadores electrónicos, y un robot manipulador equipado con una cámara de alta velocidad situada en su extremo. Además, se puede utilizar un PC como terminal para la configuración de parámetros y la visualización de señales.

El diseño del software se ha dividido en subprocesos con el fin de conseguir una mayor flexibilidad del sistema. El software de todo el sistema está codificado en VHDL (Very High speed hardware Description Language) utilizando la herramienta de diseño de software ISE Xilinx 14.7, y no se utilizó ningún procesador software. La funcionalidad de cada subproceso se ha implementado en un módulo independiente. Estos módulos están sincronizados y se ejecutan de forma paralela, aprovechando la arquitectura de la FPGA.

Dos módulos se separaron del resto (ver la FPGA en la Figura 1). Estos módulos son dependientes del hardware, pero el resto de la arquitectura propuesta no lo es. El primero es el módulo Frame Grabber. Se encarga de la lectura de los datos de imagen adquiridos por la cámara. Depende del protocolo utilizado por la cámara elegida para la aplicación de control visual (en nuestro caso este protocolo es el CameraLink). El segundo módulo dependiente del hardware es un interfaz de comunicación (protocolo ARCNET), ya que todos los comandos al controlador de los servos del robot, así como toda la información del controlador de los servos se realizan a través de ARCNET. Este módulo genera los comandos de control (en formato de paquetes de 256 bytes) en base a los datos recibidos desde el módulo de Ley de control y los envía al controlador de los servos para ser ejecutado en cada ciclo de control. Después de transmitir los comandos de control al controlador de los servos, el valor y el estado actual de cada articulación del robot se transmiten de vuelta a la FPGA, siendo interpretados por el módulo de interfaz de protocolo ARCNET.

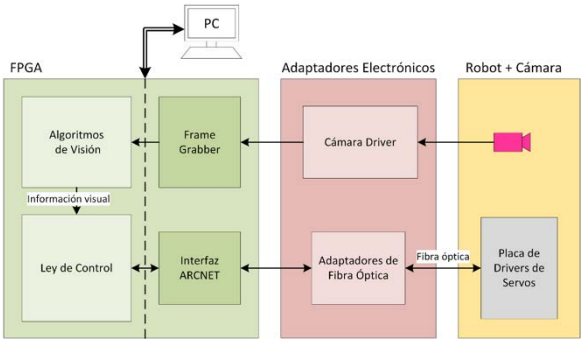

Fig. 1. Arquitectura hardware del sistema de control visual embebido.

El resto de los módulos que componen la arquitectura propuesta no son dependientes del hardware y se han dividido por su funcionalidad: Visión y Control.

El módulo de Visión consiste en un conjunto de cinco etapas. Captura las imágenes recibidas por el *Frame Grabber* para extraer los datos visuales deseados. Los algoritmos de visión están implementados en un cauce segmentado. Por lo tanto, cada píxel recibido por el módulo de visión se procesa cuando llega, sin la necesidad de almacenar toda la imagen en una memoria intermedia antes de procesarla. De esta forma, los píxeles de la imagen se procesan a la frecuencia máxima de la placa FPGA (200 MHz en la Xilinx KC705). Por lo tanto, la latencia del sistema completo se reduce, y además, se

ahorra una considerable cantidad de recursos de almacenamiento y el tráfico de comunicación será menor. La implementación usando la técnica de cauce segmentado es especialmente útil para obtener un tiempo de respuesta reducido en un tiempo conocido. Esta es la característica más importante de la arquitectura propuesta con respecto a la tradicional: CPU de propósito general. Con la arquitectura propuesta en este artículo, el sistema de control visual se convierte en un sistema de tiempo real.

La Figura 2 representa la funcionalidad de cada etapa. La primera etapa es una fase de Preprocesado. Esta etapa prepara la imagen con el fin de obtener un único formato de la imagen sin importar qué cámara se emplea (por ejemplo, conversión de la imagen de color a escala de grises, cambio de tamaño de imagen, filtrado de ruido, etc.). En la segunda etapa, la imagen se digitaliza 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. Esta es una operación morfológica sobre la imagen que elimina los píxeles situados en el borde de cualquier objeto en la imagen. Este paso se utiliza principalmente para reducir el ruido de la imagen y se puede usar 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 objetos. La función de este módulo es la detección de los píxeles conectados que forman un objeto en la imagen y asignar una etiqueta a ellos. Por último, el módulo de visión tiene que proporcionar las coordenadas en el plano imagen del centro de gravedad de las características visuales al módulo siguiente (Control). El quinto submódulo (cálculo del área y centroide) lleva a cabo esta tarea.

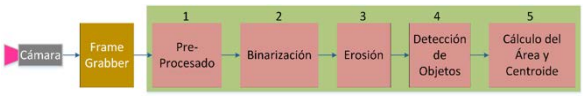

Fig. 2 Esquema del módulo de Visión.

3 CONTROL VISUAL

En este apartado se describen las principales propiedades de los sistemas de control visual directo implementados haciendo uso de la arquitectura propuesta basada en FPGA. En primer lugar, se ha implementado un sistema de control visual directo basado en la Jacobiana transpuesta. Además, se ha hecho uso del framework propuesto en [3] y uno de los controladores generados haciendo uso de este framework también se ha implementado en la arquitectura propuesta.

3.1 CONTROLADOR VISUAL BASADO EN JACOBIANA TRANSPUESTA

En este artículo se considera el uso de un sistema de visión de cámara en el extremo del robot donde $\mathbf{r}$ representa la localización del extremo del robot y $\mathbf{s} = \left[f_{1x}, f_{1y}, f_{2x}, f_{2y}, \ldots, f_{kx}, f_{ky}\right]^T \in \Re^{2k}$ es un vector de k puntos característicos extraídos de la imagen. La relación entre velocidades en el espacio imagen y $\dot{\mathbf{r}}$ viene representado por $\dot{\mathbf{s}}_r = \mathbf{L}_s \dot{\mathbf{r}}$, donde $\mathbf{L}_s$ es la matriz de interacción [1]. La velocidad de las características en el espacio imagen $\dot{\mathbf{s}}_r$ puede ser relacionada con la velocidad articular $\dot{\mathbf{q}}$ de la siguiente manera:

$$\dot{\mathbf{s}}_r = \mathbf{L}_s \mathbf{J}_r \dot{\mathbf{q}} = \mathbf{L}_J \dot{\mathbf{q}} \qquad (1)$$

donde $\mathbf{J}_r$ es la Jacobiana del manipulador y $\mathbf{L}_J$ es la Jacobiana que establece la relación entre el espacio articular y el espacio imagen. La aceleración en el espacio imagen o derivada temporal segunda de $\mathbf{s}_r$ puede obtenerse diferenciando (1) con respecto al tiempo:

$$\ddot{\mathbf{s}}_r = \mathbf{L}_J \ddot{\mathbf{q}} + \dot{\mathbf{L}}_J \dot{\mathbf{q}} \qquad (2)$$

Un sistema de control visual basado en Jacobiana transpuesta puede ser obtenido de la siguiente manera para tareas de posicionamiento [2]:

$$\boldsymbol{\tau} = \mathbf{L}_J^T \mathbf{K}_{Pt} \mathbf{e}_s - \mathbf{K}_{Dt} \dot{\mathbf{q}} + \mathbf{g} \qquad (3)$$

donde $\mathbf{K}_{Pt}$ y $\mathbf{K}_{Dt}$ son matrices proporcionales y derivativas, $\mathbf{e}_s$ es el error entre la configuración actual de las características en la imagen y las deseadas, $\mathbf{g} \in \Re^{nx1}$ es la fuerza gravitacional y $\boldsymbol{\tau} \in \Re^{nx1}$ es el vector de pares aplicados al robot (n es el número de grados de libertad del robot).

3.2. CONTROLADOR VISUAL BASADO EN LA INVERSA DEL CUADRADO DE LA MATRIZ DE INERCIA

En este apartado se hará uso del framework descrito en [3]. La función de control que minimiza los pares aplicados a un robot articular cumpliendo una determinada restricción viene definida por la siguiente función:

$$\boldsymbol{\tau} = \mathbf{W}^{-1/2}\left(\mathbf{A}\mathbf{M}^{-1}\mathbf{W}^{-1/2}\right)^{+} \cdot \left(\mathbf{b} - \mathbf{A}\mathbf{M}^{-1}\left(-\mathbf{C} - \mathbf{g}\right)\right) \qquad (4)$$

donde $\mathbf{M} = \mathbf{M}(\mathbf{q}) \in \Re^{nxn}$ es la matriz de inercia del manipulador (simétrica definida positiva), $\mathbf{C} = \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \in \Re^{nx1}$ representa el vector de fuerzas centrípetas y de Coriolis, y $\mathbf{W}$ es una función dependiente del tiempo a ajustar. El símbolo +

representa la pseudo-inversa de una matriz general y las m restricciones de la tarea a realizar vienen dadas por:

$$\mathbf{A}(\mathbf{q},\dot{\mathbf{q}},t)\ddot{\mathbf{q}} = \mathbf{b}(\mathbf{q},\dot{\mathbf{q}},t) \quad (5)$$

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

El objetivo principal del sistema de control visual es guiar el robot manipulador a lo largo de una trayectoria definida en el espacio imagen como referencia. Por lo tanto, la descripción de la tarea puede venir dada por la siguiente ecuación en el espacio imagen:

$$(\ddot{\mathbf{s}}_d - \ddot{\mathbf{s}}) + \mathbf{K}_D(\dot{\mathbf{s}}_d - \dot{\mathbf{s}}) + \mathbf{K}_P(\mathbf{s}_d - \mathbf{s}) = 0 \quad (6)$$

donde $\ddot{\mathbf{s}}_d$, $\dot{\mathbf{s}}_d$ y $\mathbf{s}_d$ son variables en el espacio imagen que representan las aceleraciones, velocidades y posición respectivamente. $\mathbf{K}_P$ y $\mathbf{K}_D$ son las matrices de constantes proporcional y derivativa, respectivamente. Esta ecuación puede ser expresada de la siguiente manera:

$$\ddot{\mathbf{s}}_d + \mathbf{K}_D\dot{\mathbf{e}}_s + \mathbf{K}_P\mathbf{e}_s = \ddot{\mathbf{s}}_r \quad (7)$$

donde $\mathbf{e}_s$ y $\dot{\mathbf{e}}_s$ son el error en imagen y la derivada con respecto al tiempo de dicho error, respectivamente. La variable $\ddot{\mathbf{s}}_r$ representa la referencia expresada como aceleraciones en el espacio imagen. Reemplazando esta última variable con el valor obtenido en (2) se puede obtener la siguiente expresión:

$$\ddot{\mathbf{s}}_d + \mathbf{K}_D\dot{\mathbf{e}}_s + \mathbf{K}_P\mathbf{e}_s - \dot{\mathbf{L}}_J\dot{\mathbf{q}} = \mathbf{L}_J\ddot{\mathbf{q}} \quad (8)$$

Ahora es posible expresar la tarea de control visual en la forma de las restricciones establecidas en la expresión (5). Para ello, las matrices **A** y **b** adquieren el siguiente valor:

$$\begin{aligned} \mathbf{A} &= \mathbf{L}_J \\ \mathbf{b} &= \ddot{\mathbf{s}}_d + \mathbf{K}_D\dot{\mathbf{e}}_s + \mathbf{K}_P\mathbf{e}_s - \dot{\mathbf{L}}_J\dot{\mathbf{q}} \end{aligned} \quad (9)$$

A continuación es necesario definir el valor de la matriz de pesos **W**. Esta matriz define el comportamiento dinámico del controlador durante el seguimiento de la trayectoria deseada en el espacio imagen. En [3] se obtuvieron buenos resultados empleando el valor de $\mathbf{W} = \mathbf{M}^{-2}$. Considerando este valor, así como los valores indicados en la Ecuación (9), el valor final del controlador será el siguiente:

$$\boldsymbol{\tau} = \mathbf{M}\mathbf{L}_J^{+}\left(\ddot{\mathbf{s}}_d + \mathbf{K}_D\dot{\mathbf{e}}_s + \mathbf{K}_P\mathbf{e}_s - \dot{\mathbf{L}}_J\dot{\mathbf{q}}\right) + \mathbf{C} + \mathbf{g} \quad (10)$$

En el siguiente apartado se describirán los resultados obtenidos tras la implementación de ambos controladores en la arquitectura propuesta basada en FPGA.

4 RESULTADOS

En el siguiente apartado se muestran dos experimentos que emplean los dos controladores definidos en la Sección 3 para realizar una tarea de control visual basada en imagen. En ambos casos se hace uso de la arquitectura basada en FPGA. En el caso del controlador basado en Jacobiana transpuesta se trata de una tarea de posicionamiento. Sin embargo, en el caso del controlador basado en el framework de control óptimo el sistema realizará el seguimiento de una trayectoria en el espacio imagen. Con el objetivo de obtener resultados similares en ambos casos, la trayectoria deseada en la imagen para este segundo controlador será rectilínea entre la configuración inicial de las características en la imagen y la configuración deseada. El robot manipulador utilizado es un Mitshubishi PA10 de 6 grados de libertad.

En la Figura 3 se ha representado la evolución de las características en la imagen durante la ejecución de ambos controladores. Se observa que en ambos casos las características siguen una trayectoria recta entre la posición inicial y final, alcanzando en ambos casos correctamente la posición deseada en la imagen.

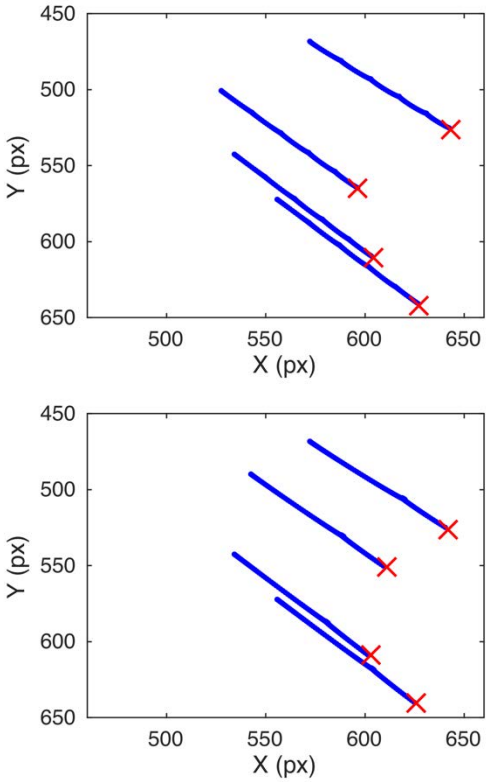

Figura 3: Trayectorias de las características en la imagen para: a) Controlador de Jacobiana Transpuesta; b) Controlador basado en la inversa del cuadrado de la matriz de inercia.

Con el propósito de observar el comportamiento en el espacio tridimensional, a continuación se va a estudiar cómo evoluciona el error en traslación y en

rotación (roll, pitch, yaw) entre la posición 3D inicial y la deseada. Para ello, en la Figura 4 se representa la evolución de ambos errores durante la ejecución de los dos controladores propuestos en el Apartado 3. Ambos errores acaban anulándose, sin embargo, en el espacio 3D se observa un mejor comportamiento cuando el controlador implementado en la FPGA es un controlador basado en el framework de control óptimo.

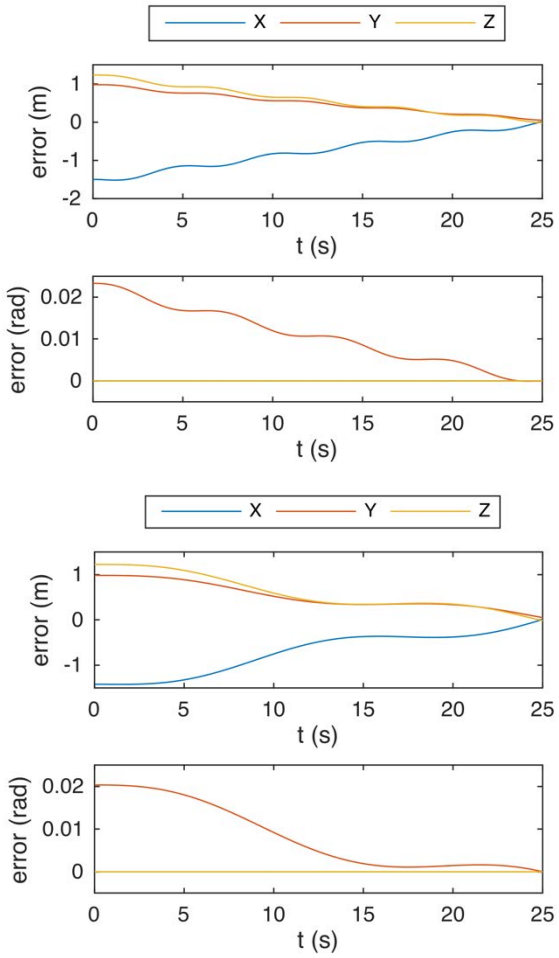

Figura 4: Error en traslación y rotación obtenido durante la ejecución para: a) Controlador de Jacobiana Transpuesta; b) Controlador basado en la inversa del cuadrado de la matriz de inercia.

Por último se van a analizar las acciones de control generadas por los controladores. Dichas acciones de control se han representado en la Figura 5. Esta última figura representa los pares generados por los controladores para alcanzar la configuración final a partir de la inicial.

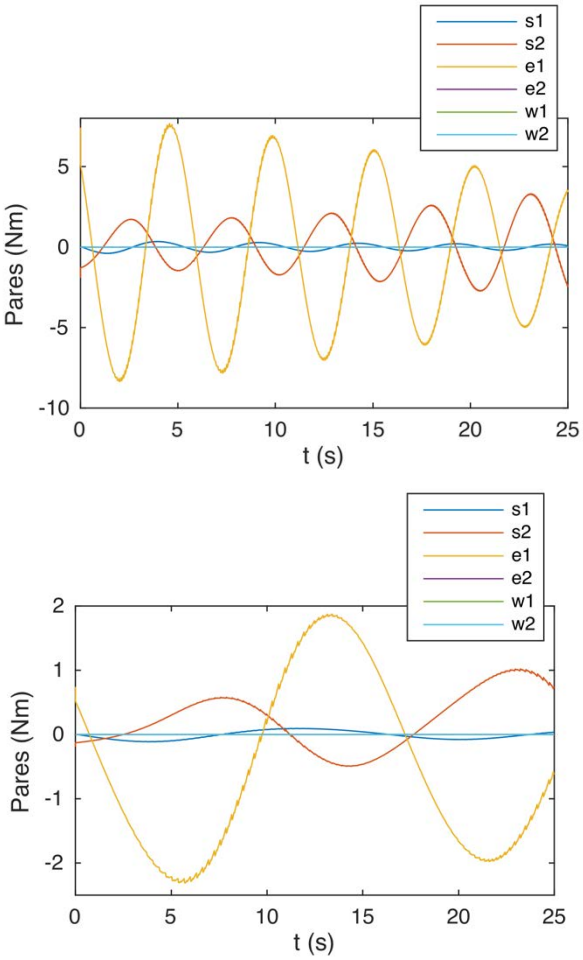

Figura 5: Acciones de control generadas por: a) Controlador de Jacobiana Transpuesta; b) Controlador basado en la inversa del cuadrado de la matriz de inercia.

Un factor importante a tener en cuenta en la implementación de cualquier software embebido en una FPGA es la ocupación de sus recursos lógicos. La Tabla 1 muestra un resumen de los recursos empleados por la FPGA Kintex-7 de Xilinx utilizada en este trabajo. El alto requerimiento de recursos lógicos para la implementación de los algoritmos de control se debe fundamentalmente al gran número de operaciones matriciales requeridas por los controladores (multiplicación, inversión de matrices, etc.), las dimensiones de las matrices son de 8x8 como mucho y sus elementos se expresan en el formato de punto flotante de doble precisión especificado en el estándar IEEE-754. La fila “otros” de la Tabla 2 muestra los recursos consumidos para implementar otras funciones que no se describen en este artículo, como el módulo de comunicación para el interfaz que conecta la FPGA con el PC y el controlador de HDMI que permite mostrar las imágenes capturadas y procesadas en un monitor externo.

Tabla 1: Uso de los recursos de la FPGA.

	Puertas lógicas	**LUTs**	**LUT-RAM**	**Bloques RAM/FIFO**	**Bloques DSP**
Algoritmos de Visión	327	7865	400	49	6
Algoritmos de Ley de control	6525	19511	411	0	17
Otros	1572	4572	15	0	0
Total	10834	31948	826	49	23
Ocupación de la FPGA	21%	15%	1%	4%	2%

CONCLUSIONES

Los sistemas de control visual directo requieren un sistema de visión rápido que sea capaz de proporcionar una nueva referencia para el controlador en el menor tiempo posible. El sistema que se ha propuesto en este artículo está implementado sobre una FPGA. Las posibilidades de paralelización de los algoritmos de visión y control que ofrece una FPGA no sólo incrementan la velocidad de estos sistemas de control visual directo, sino que además lo convierten en un sistema de tiempo real, ya que permiten fijar el tiempo en el que se tendrá una nueva referencia visual para el controlador, así como el tiempo en el que se tendrá una nueva acción de control. El sistema propuesto se ha verificado implementando sobre la FPGA dos controladores directos distintos.

Agradecimientos

Este trabajo ha sido financiado por el Ministerio de Economía y competitividad mediante el proyecto DPI2015-68087-R.

Referencias

[1] Chaumette, F.; Hutchinson, S., (2006) “Visual Servo Control, Part I: Basic Approaches”, *IEEE Robotics and Automation Magazine*, vol. 13, no. 4, pp. 82-90, Dec.

[2] Kelly, R.; Carelli, R.; Nasisi, O.; Kuchen, B.; Reyes, F., (2000) “Stable Visual Servoing of Camera-in-Hand Robotic Systems”. *IEEE/ASME Transactions on Mechatronics*, vol. 5, no. 1, pp. 39-48, March.

[3] Jara, C. A.; Pomares, J.; Candelas, F. A.; Torres, F., (2014) “Control Framework for Dexterous Manipulation Using Dynamic Visual Servoing and Tactile Sensors’ Feedback”. *Sensors*, vol. 14, no. 1, pp. 1787-1804, Jan.

[4] Nguyen, K. Q.; Nguyen, T. H.; Ha, Q. P., (2014) "FPGA-Based Sensorless PMSM Speed Control Using Reduced-Order Extended Kalman Filters" *IEEE Transactions on Industrial Electronics*, vol.61, no.12, pp.6574,6582, Dec.

[5] Roshandel, N. T.; Dinavahi, V., (2015) "A General Framework for FPGA-Based Real-Time Emulation of Electrical Machines for HIL Applications," *IEEE Transactions on Industrial Electronics*, vol. 62, no.4, pp.2041-2053, April.

[6] Wei, L.; Gregoire, L. A.; Belanger, J., (2015) "A Modular Multilevel Converter Pulse Generation and Capacitor Voltage Balance Method Optimized for FPGA Implementation," *IEEE Transactions on Industrial Electronics*, vol.62, no.5, pp.2859-2867, May.

[7] Vachhani, L.; Sridharan, K.; Meher, P. K., (2009) "Efficient FPGA Realization of CORDIC With Application to Robotic Exploration," *IEEE Transactions on Industrial Electronics*, vol.56, no.12, pp.4915-4929, Dec.

[8] Shao, X.; Sun, D.; Mills, J. K., (2006) “A New Motion Control Hardware Architecture with FPGA-Based IC Design for Robotic Manipulators”. *Proc. IEEE International Conference on Robotics and Automation*, pp. 3520-3525.

[9] Chang, T. N.; Biao, C.; Sriwilaijaroen, P., (2006) "Motion Control Firmware for High-Speed Robotic Systems," *IEEE Transactions on Industrial Electronics*, vol.53, no.5, pp.1713-1722, Oct.

[10] Zhang, L.; Slaets, P.; Bruyninckx, H., (2012) “An open embedded hardware and software architecture applied to industrial robot control”, *Proc IEEE International Conference on Mechatronics and Automation*, pp. 1822–1828

[11] Lee, W. K.; Jung, S., (2006) "FPGA Design for Controlling Humanoid Robot Arms by Exoskeleton Motion Capture System", *Proc of the IEEE International Conference on Robotics and Biomimetics*, pp. 1378 – 1383.

[12] Wen-Hong, Z.; Lamarche, T.; Dupuis, E.; Jameux, D.; Barnard, P.; Liu, G., (2013) "Precision Control of Modular Robot Manipulators: The VDC Approach With Embedded FPGA". *IEEE Trans. on Robotics*, vol. 29, no. 5, Oct.

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

[14] 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., (2011) "FPGA-based Embedded Visual Servoing Platform for Quick Response Visual Servoing" *Proc Asian Control*, pp. 263-268.

[15] Pomares, J.; Perea, I.; Torres, F., (2014) "Dynamic Visual Servoing With Chaos Control for Redundant Robots," *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 2, pp 423-431.

[16] Pomares, J.; Garcia, G. J.; Perea, I.; Corrales, J. A.; Jara, C. A.; Torres, F., (2011) "Visual control of a multi-robot coupled system: Application to collision avoidance in human-robot interaction" *Proc. IEEE International Conference on Computer Vision*, pp.1045-1051.