Desarrollo de un sensor de visión para aprendizaje por refuerzo en robótica

Rocío Guasch[1], Sergio Cuenca[2], Tomás Martínez[1]

[1] Depto. Física, Ingeniería de Sistemas y Teoría de la Señal, Universidad de Alicante{tomas@dfists.ua.es}

[2] Depto. Tecnología Informática y Computación, Universidad de Alicante {sergio@dtic.ua.es}

Resumen— En este trabajo se describe el proceso de desarrollo de un sensor que incluye los algoritmos necesarios para realizar el control de servomotores mediante realimentación visual. El objetivo final que se persigue, es obtener un sensor de bajo coste y rendimiento suficiente para integrarse en un sistema de navegación basado en aprendizaje por refuerzo. La utilización de un lenguaje de descripción hardware de alto nivel, junto con el desarrollo de un entorno de verificación basado en PC, han sido los puntos clave en el éxito del trabajo.

Palabras clave—FPGA, Visual Servoing, Aprendizaje por Refuerzo.

I. INTRODUCCIÓN

El aprendizaje por refuerzo (Reinforcement Learning o RL) [1] intenta simular en un dispositivo electrónico la forma en que las personas aprendemos y actuamos en relación a las experiencias que vivimos. El proceso consiste, básicamente, en aprender a decidir, ante una situación determinada, que acción es la más adecuada para lograr un objetivo. Este método es adecuado cuando no existe un conocimiento "a priori" o éste es demasiado complejo como para utilizar otras técnicas, por lo que presenta indudables ventajas en robótica móvil.

Los sensores de imagen, por otro lado, juegan un papel cada vez más importante en estos sistemas [2]. Las cámaras digitales son pequeñas, ligeras y tienen un consumo muy reducido, sin embargo el principal escollo para su utilización estriba en la gran cantidad de datos que es necesario procesar para obtener información relevante con la que alimentar a los sistemas de navegación de los robots. Para solventar este problema se han propuesto diversas soluciones: diseño de ASICs especializados [3], sistemas basados en DSPs, diseño de sistemas on chip sobre FPGAs [4] y sistemas híbridos FPGA/CPU [5]. En la mayoría de los casos se trata de obtener el máximo rendimiento a costa de incrementar notablemente el coste del sistema final.

En este trabajo proponemos la utilización de FPGAs en el desarrollo de un sensor de visión que sea fácilmente integrable en un robot móvil de bajo coste. Este robot utiliza RL para la navegación autónoma mediante la identificación de marcas en el entorno. El desarrollo está enfocado a obtener el rendimiento mínimo necesario, con el mínimo de recursos y procurando una resolución suficiente en las imágenes que se procesan.

El presente documento trata de recoger y explicar el trabajo desarrollado para la implementación de este sensor, las herramientas que han sido necesarias para la ejecución de todos los programas, las dificultades presentadas durante su implementación, las pruebas realizadas y los resultados obtenidos.

A. Visual Servoing

Visual Servoing (VS) engloba todas las técnicas de control de robots (tanto manipuladores como robots móviles) que utilizan realimentación visual [6]. El sensor de visión puede estar fijado en el entorno o puede ir incorporado en el robot, como es el caso de la visión en los seres vivos. Además, dicho sensor puede ser estático con respecto al robot o dinámico incorporando sus propios grados de libertad, tal como sucede en el ojo humano. En este trabajo se realizará el control visual de un robot móvil no holónomo mediante un sensor de visión situado en una torreta móvil que dispone de dos grados de libertad.

Desde el punto de vista del lazo de realimentación las distintas técnicas de VS se pueden clasificar en *control visual basado en posición* (PBVS, position based visual servoing), en *control visual basado en imagen* (IBVS, image based visual servoing), o bien adoptar un enfoque híbrido (HBVS) [7]. Habitualmente el método PBVS utiliza la imagen capturada para estimar la posición y velocidad del objeto de interés en tres dimensiones (3D). Para ello requiere un considerable tratamiento analítico a través de transformaciones matriciales y estimación del Jacobiano. Por tanto, necesita un modelo del robot y su calibración previa. En cambio, la técnica IBVS utiliza directamente parámetros extraídos de la imagen para cerrar el lazo de control. De este modo, se puede llevar a cabo el control visual de un robot sin modelo y con una mínima calibración. En contrapartida, el control resulta más complejo, ya que la imagen introduce principalmente no linealidades que dificultan la aplicación de técnicas de control lineal clásico.

El método IBVS será el utilizado en este trabajo, ya que dentro del enfoque de aprendizaje por refuerzo el objetivo es planificar el movimiento del vehículo sin utilizar un modelo previo ni calibración de los sensores, entre ellos el sensor de visión. Este método se ha aplicado al control visual de robots móviles holónomos [8]. También ha sido aplicado a vehículos no holónomos [9], empleando un PC externo para procesar la imagen y cerrar los lazos de control. En cambio, en este trabajo se propone incorporar el procesado de imagen y el control de bajo nivel en una FPGA.

B. Aprendizaje por refuerzo

Los métodos de aprendizaje por refuerzo solo requieren una recompensa escalar (o penalización) para aprender a asociar situaciones (estados) con acciones [8]. Al contrario que en aprendizaje supervisado, no se necesita un profesor (conjunto de patrones de entrenamiento) para adquirir un comportamiento

óptimo. Solamente se requiere interactuar con el entorno aprendiendo por experiencia. El conocimiento se almacena en una tabla que contiene una estimación de la recompensa acumulada para alcanzar el estado objetivo desde cualquier estado inicial. Las acciones óptimas son aquellas que proporcionan la máxima recompensa en cada estado.

El método más popular de aprendizaje por refuerzo es Q-learning, debido a que su formulación es simple y además resuelve problemas sin crear un modelo durante la interacción con el entorno. La recompensa acumulada de cada par estado-acción *Q(s,a)* se actualiza en cada iteración mediante la ecuación siguiente

$$\Delta Q(s,a) = \alpha\left(r + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s,a) \right)$$

donde *Q(s,a)* es el valor esperado al realizar la *a* en el estado *s*, *r* es la recompensa, α es el factor de aprendizaje que controla la convergencia del algoritmo y γ es el factor de descuento. El factor de descuento da más valor a las recompensas obtenidas en estados cercanos que a las correspondientes a estados más alejados del estado actual. Si la función recompensa satisface ciertas propiedades, el factor de descuento puede ser omitido (γ = 1). La acción *a* con el valor *Q* mayor en el estado *s* es la mejor acción hasta el instante *t*.

En aplicaciones con sistemas reales Q-learning emplea demasiado tiempo haciendo cientos de miles de intentos para aproximar el comportamiento óptimo. Para acelerar el aprendizaje es necesario incorporar algún mecanismo de planificación. Prioritized sweeping y Dyna-Q incluyen un mecanismo de búsqueda para simular las experiencias reales pasadas en un orden especificado. De este modo, el modelo del sistema es almacenado mediante transiciones de estados con sus recompensas asociadas. Por las razones expuestas anteriormente, en este trabajo se almacenarán las transiciones entre estados junto con sus respectivas recompensas (en realidad, penalizaciones en forma de tiempo negativo), las cuales constituyen el modelo dinámico del sistema. Así podremos propagar miles de valores *Q(s,a)* por segundo mediante simulación.

II. Aplicación al control visual de un vehículo

La figura 1 muestra la parte delantera del vehículo que se pretende controlar. En ella se puede apreciar la torreta móvil, donde se halla instalada la cámara que proporcionará las imágenes del escenario en el que se debe desenvolver el vehículo.

El objetivo es que el sensor de visión reconozca un objeto definido, la marca de dos lóbulos cuadrados, y sea capaz de tenerlo centrado en su campo de visión en todo momento. Para que el vehículo realice esta tarea, se pretende utilizar el aprendizaje por refuerzo.

El seguimiento del objeto en lazo cerrado, se realiza mediante un controlador Proporcional Derivativo (PD), encargado de la rotación de la torreta móvil. El cálculo de la salida del controlador PD se realiza mediante la suma de los términos Proporcional y Derivativo. Si definimos la variable de control como *u(t)*, la ecuación es la siguiente:

$$u(t) = K_P e_{IM}(t) + K_D \frac{de_{IM}(t)}{dt}$$

En nuestro caso, la variable de control se codifica como una señal PWM que actúa sobre el servo de radiocontrol que acciona la torreta del vehículo. Los parámetros K_P y K_D se obtienen experimentalmente de tal modo que el sistema controlado presente un tipo de comportamiento críticamente amortiguado (coeficiente de amortiguamiento aproximadamente igual a 1). La variable $e_{IM}(t)$ es el error de posición del objeto con respecto al centro de la imagen. Dicha variable se obtiene mediante el procesado de la imagen, tal como veremos en el apartado siguiente.

Una vez centrada la marca por el sensor, obtenemos un ángulo *α* que define el ángulo que giró la torreta para conseguir centrar la marca (Fig. 1).

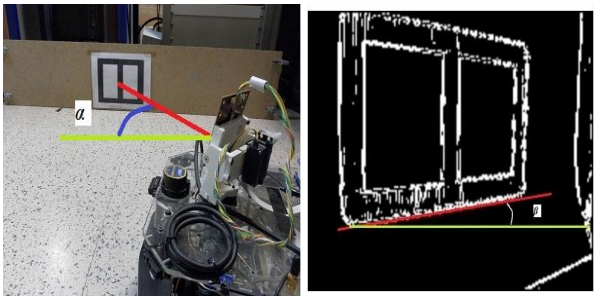

Fig. 1. Representación de las variables de estado *α* y *β*.

La segunda variable *β*, se obtiene directamente de la imagen y es proporcional a la desviación del vehículo respecto de la marca. Se calcula como el ángulo formado por el segmento inferior de la marca respecto de la horizontal (Fig. 1). Estas dos variables definen el espacio de estados bidimensional en el que se moverá el vehículo.

El objetivo es que el vehículo alcance el estado (*α, β*) = (0,0), lo que indica que el vehículo se encuentra en línea recta con la marca y que el sensor está alineado con la misma (Fig. 2). Una vez llegados a este estado, el vehículo se movería en línea recta hasta alcanzar la marca.

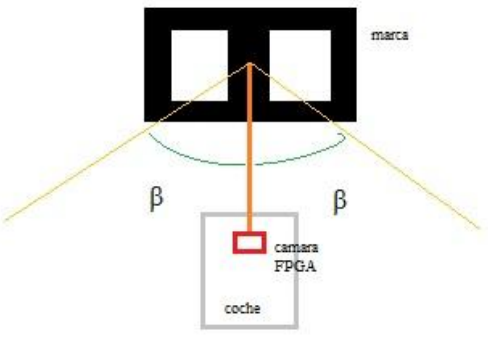

Fig. 2. Estado objetivo Alfa=beta=0

Para controlar el movimiento se emplea el aprendizaje por refuerzo como método alternativo de control no lineal (alternativo a otros métodos lineales que son más costosos en tiempo y menos eficientes). De esta forma, el vehículo se moverá solo por una zona de trabajo delimitada por las restricciones del sistema, en este caso el ángulo máximo de giro de la torreta α=±80º y desviación máxima que permite al sensor ver el objeto β=±80º. En el proceso de entrenamiento, que se lleva a cabo moviendo el vehículo por la zona de trabajo, se actualiza la tabla Q, que almacena la recompensa acumulada de cada estado hasta el objetivo. Una vez finalizado el tiempo asignado al entrenamiento el vehículo se moverá de forma óptima desde cualquier posición del espacio de trabajo hasta el objetivo. Mientras siga trabajando en modo "normal" la tabla Q se seguirá actualizando para contemplar cambios no previstos en el entorno o en la dinámica del vehículo (holguras en la mecánica de la torreta, deslizamiento de ruedas, desajustes de la imagen, cambios de iluminación, etc…).

III. Arquitectura del sistema e Implementación

La arquitectura del sensor de visión se ha diseñado teniendo en cuenta dos objetivos principalmente. En primer lugar, conseguir una implementación final de muy bajo coste. Esto supone la selección de componentes baratos tanto como la reducción al mínimo del número de estos. Por ende, esta reducción se traducirá en un menor consumo y una mayor autonomía del vehículo. En segundo lugar, obtener un rendimiento suficiente para incorporar la información visual al lazo de control del vehículo. Para la consecución de estos objetivos, en principio contrapuestos, se propone la utilización de FPGAs en la implementación del sensor, así como la realización de un estudio previo en el que se establezcan los requisitos mínimos del sistema. El establecimiento de estos requisitos permitirá ajustar al máximo las necesidades de lógica programable y la utilización de familias de bajo coste.

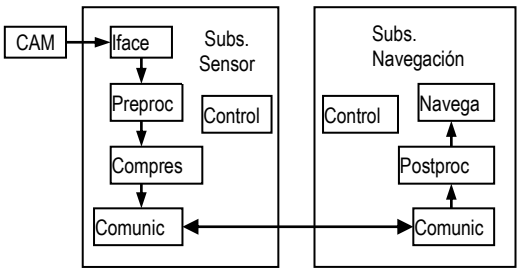

Figura 3. Esquema general del sistema completo

La figura 3 muestra la arquitectura general del sistema, que consta de dos bloques principales. El *Subsistema Sensor (SS)* integra todas las funcionalidades necesarias para adquirir y procesar las imágenes provenientes de una cámara CMOS. Este sistema debe ser capaz de transformar una importante cantidad de datos procedentes de las imágenes en un conjunto de *meta-datos* que condensen la información relevante para el control visual, reduciendo de esta forma el ancho de banda en la comunicación con el resto del sistema. En caso de ser necesario también puede incorporar un bloque de compresión para reducir aún más el número de transferencias por imagen.

El *Subsistema de Navegación (SN)*, por su parte, es el encargado de interpretar los *meta-datos* y tomar las decisiones necesarias para dirigir la navegación del vehículo aplicando el aprendizaje por refuerzo. Ambos bloques se comunican a través de un canal de bajo coste (y bajo ancho de banda) de tipo serie.

Además de las restricciones que imponen los objetivos mencionados anteriormente, hay que tener en cuenta que el sensor de visión debe integrase en un vehículo de reducidas dimensiones cuyo sistema de navegación está soportado por una tarjeta basada en el microprocesador PowerPC555 (@40MHz), dónde los únicos puertos de comunicación disponibles son RS232 y SPI. El *SN* incluye unos algoritmos de extracción de los *meta-datos*, básicamente filtros de preprocesamiento, etiquetado de *blobs*, identificación de la marca y cálculo de su posición. El sistema original tiene un rendimiento inferior a 2fps, insuficiente para la mayoría de las aplicaciones que se pretenden abordar con este prototipo.

Para establecer los requisitos del *SS* se realizaron diversos experimentos utilizando un simulador "ad-hoc" y secuencias de imágenes tomadas directamente por la cámara seleccionada para el sistema final (CMOS 1,5MPix). Como conclusión del estudio se seleccionaron los filtros de procesamiento que debían incluirse, así como la precisión de las operaciones, el mínimo *framerate* y el tamaño y profundidad de las imágenes. Los resultados pueden observarse en la tabla I.

TABLA I

REQUISITOS MÍNIMOS DEL SS

framerate	Resolución	Bits/pix	Filtro1	Filtro2
12fps	640x480	8bpp	SobelXY (3x3)	Open (3x3)

Inicialmente se consideró incorporar sólo los filtros de preprocesamiento en la FPGA, para reducir el tiempo de desarrollo. Esto implica un ancho de banda aproximado para la transmisión de las imágenes preprocesadas de 3,6MB/s, que está por encima de la capacidad de los buses de comunicación disponibles en la tarjeta.

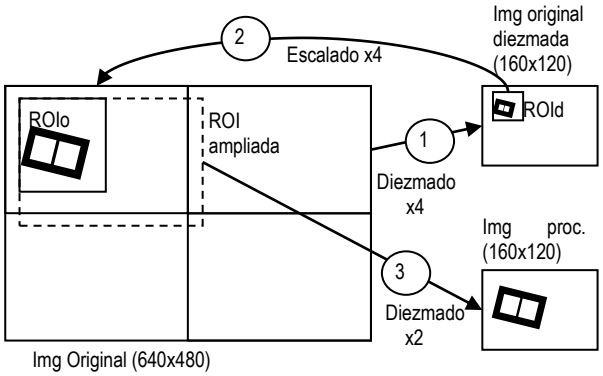

Figura 4. Proceso de diezmado

En consecuencia, se optó por reducir el tamaño de las imágenes a transmitir pero conservando, en la medida de lo posible, la resolución inicial. Esto se consigue trabajando sobre la imagen completa pero transmitiendo sólo una región de interés (*ROI*) de tamaño fijo 160x120. En el caso de la *ROI* sea mayor de este

tamaño, se realiza un diezmado de la imagen manteniendo la máxima resolución posible, de forma que la pérdida de precisión en los resultados sea la mínima.

Los casos más extremos se producen cuando la *ROI* abarca la totalidad de la imagen, siendo necesario un diezmado x4 en ambas dimensiones. No obstante esto ocurre sólo en dos tipos de casos; en los estadios iniciales de la navegación, cuando el vehículo se encuentra muy lejos de la marca y es necesario explorar el campo de visión completo, o en los últimos movimientos próximos a la marca, cuando ésta ocupa una fracción importante del campo de visión. En el primer caso, la pérdida de precisión no es importante ya que se trata de identificar, groso modo, la marca en la imagen diezmada para seguidamente centrar la *ROI* a la máxima resolución. En el segundo caso, el diezmado no disminuye significativamente la precisión en el cálculo de la posición de la marca ya que se dispone de un número de importante de píxeles para identificarla.

En la figura 4, puede verse el proceso de cálculo del diezmado que lleva a cabo el Subsistema de navegación. Inicialmente se cuenta con la imagen original diezmada x4 (1) sobre la que se realiza la primera identificación de la marca. A continuación escoge la zona de interés (ROId) y se calculan las coordenadas de la ROI original (ROIo) (2). Estas coordenadas se transmiten al *SS* que, a su vez, calcula el nivel de diezmado y amplia el ROI, si fuera necesario, para completar la imagen a transmitir (3).

Las modificaciones se incorporaron al simulador obteniéndose unas desviaciones aceptables en los resultados respecto del sistema original.

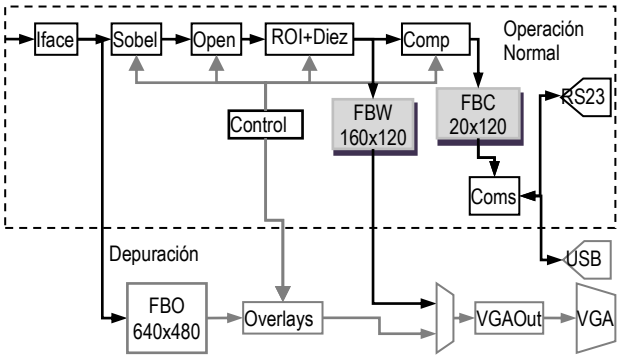

Fig. 5. Esquema del subsistema sensor

La figura 5 muestra el esquema del *SS*. Los bloques utilizados durante la operación normal del sensor están agrupados en el módulo superior (línea discontinua). En él se incluye, además, un bloque de compresión que garantiza un adecuado *framerate* incluso utilizando el puerto RS232. Este bloque agrupa 8 pixels binarios (1bit por pixel) en una sola palabra de 8bits, con lo que se consigue una compresión sin pérdidas de 8 a 1. También se dispone de dos *framebuffers*, en los que se almacena la *ROI* de trabajo (FBW) y la *ROI* comprimida (FBC). El bloque de Comunicaciones, permite tanto la transmisión de datos y estado hacia el *SN*, como la transmisión de comandos y parámetros de configuración hacia el *SS*. En la tabla II se muestran las diferentes posibilidades de configuración para el *SS*.

Para la realización de un prototipo se utilizó la tarjeta RC10 de Celoxica que incluye una cámara CMOS de 1,5Mpixels y una FPGA *Spartan3 1500L* . El diseño completo se codificó con el lenguaje *HandelC* [10], y la librería de procesamiento de imagen PixelStream [11], lo que aceleró significativamente el desarrollo y facilitó la depuración. La implementación se llevó a cabo con la herramienta *ISEv10.1* de Xilinx [12]. En el modo normal de operación el sistema no utiliza memoria externa y los framebuffers están constituidos enteramente por memoria de bloques de la FPGA. Los recursos utilizados son 3.819 Slices: (18%), 21 BRAM: (65%), lo que permitiría acomodar el diseño en un dispositivo más modesto o utilizar la lógica restante para incorporar una mayor parte del procesamiento en la FPGA. La máxima frecuencia de funcionamiento que reporta la herramienta es de 61,3MHz.

TABLA II

PARÁMETROS DE CONFIGURACIÓN DEL SS

Parámetro	Tipo	Descripción
X0,Y0	signed 16	Coordenadas de la esquina superior izquierda del ROI
X1,Y1	signed 16	Coordenadas de la esquina inferior derecha del ROI
ProcCtrl	unsig 2	Control del pipe de procesamiento: Nproc, Sobel, Sobel+Open.
CompCtrl	unsig 2	Control del tipo de compresión
Threshold	unsig 8	Umbral de segmentación Sobel
ScaleLevel	unsig 2	Nivel de diezmado : $2^{ScaleLevel}$

El sensor puede trabajar máximo *framerate* de la cámara (50fps) siendo la transmisión vía serie la única limitación en este sentido.

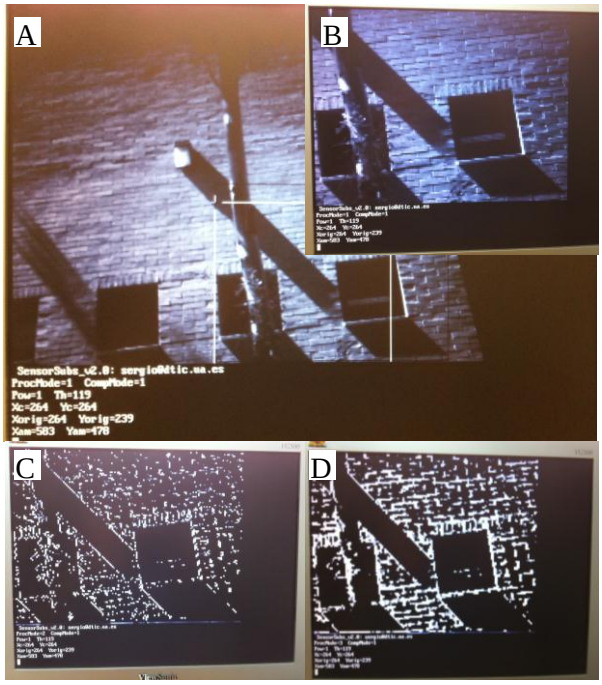

Fig. 6. Modos de visualización/procesamiento en modo depuración

Para ayudar a la depuración del diseño se incluyeron varios bloques adicionales que se muestran en color gris en la figura 5. También se añadió a la tarjeta una memoria tipo SSRAM que hace el papel de *framebuffer* externo para almacenar la imagen en su tamaño original. Cuando el sistema trabaja en modo depuración, el sensor genera señal de video permitiendo visualizar, en un monitor, la imagen original o la ROI. En la figura 6 se pueden apreciar varios de los modos de visualización y procesamiento disponibles durante la depuración:

a) Imagen completa (640x480), la línea blanca delimita la *ROI* original y la negra la *ROI* ampliada.

b) Imagen de trabajo (160x120) sin procesar. En este caso las dimensiones de la ROI están entre 160x120 y 320x240, por lo que es necesario realizar un diezmado x2 en la ROI ampliada para obtener la imagen de trabajo.

c) Imagen de trabajo procesada con filtro de sobel y umbral 116.

d) Imagen de trabajo procesada con filtro de sobel y apertura 3x3.

En todas las imágenes se superpone un *overlay* de consola de texto en la que se muestran en tiempo real los parámetros de configuración: tipo de procesamiento, valor del umbral, coordenadas del ROI, etc…

IV. INTEGRACIÓN

Para facilitar la integración del sensor en el vehículo, se desarrolló una aplicación en *LabWindows CVI 8.0* de National Instruments [13]. La aplicación se comunica con el sensor a través de RS232 (o USB) utilizando un sencillo protocolo que permite sincronizar el envío/recepción de comandos/datos en ambas direcciones. A través de una interfaz gráfica el usuario puede visualizar las imágenes transmitidas por el sensor e interaccionar con él mediante distintos controles (botones, cajas de texto, etc…) para configurar sus parámetros (Fig. 7). El modo de depuración del sensor permite comprobar que las imágenes y los comandos transmitidos a través de la interfaz son correctos.

La aplicación también cuenta con una versión ejecutable del código del *SN*, de forma que es posible validar el sistema completo en lazo abierto. A lo largo de todo el periodo del proyecto se han ido realizando una serie de pruebas que han servido para probar y mejorar el software implementado. A continuación se describen los aspectos más relevantes.

A. Procesado de imagen para determinar la posición de la marca

Con el fin de que el sistema sea más robusto frente a cambios en la iluminación, fondo, etc.., la detección de la marca se realiza mediante la detección de sus lóbulos en lugar de utilizar las líneas que la definen. El procesado comienza con el etiquetado de los objetos presentes en la imagen, donde se asigna a cada zona un color (figura 7). A continuación, se realiza un histograma de la imagen lo que nos proporciona información sobre el tamaño de los objetos etiquetados. Utilizando como referencia el tamaño de la marca respecto del tamaño de la imagen total (relación conocida a priori) y ecualizando el histograma, se realiza una primera selección de las zonas candidatas a representar la marca.

Para poder discriminar entre los objetos, se calculan diversas características: media, desviación típica y distancia euclídea. Aquella pareja de objetos que presente las características más similares serán los candidatos a formar parte de la marca.

Utilizando esta información se procede a calcular el punto medio entre ellos, así como los números de la última columna del lóbulo izquierdo y de la primera columna del lóbulo derecho (hueco). En la figura 7 se muestra la imagen etiquetada una vez que se ha ecualizado el histograma. Como puede apreciarse solo quedan 4 objetos y dos de ellos con características muy similares. La línea central entre ellos se ha dibujado de color verde.

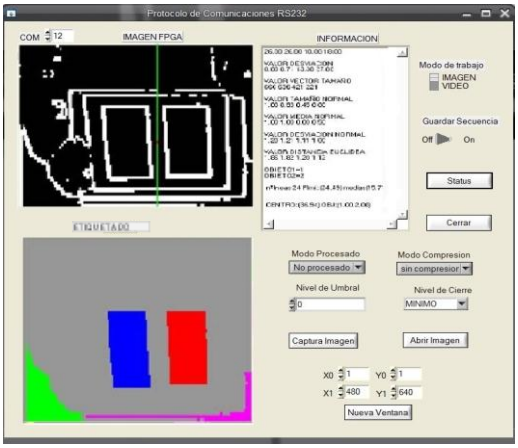

Fig. 7. Interfaz de depuración construido con CVI.

En la figura 8, muestra otro ejemplo donde la marca se encuentra más alejada de la cámara y la ecualización en este caso ha generado cinco objetos. De nuevo la discriminación ha identificado a los dos lóbulos de la marca como los dos objetos más similares y se ha calculado correctamente la línea central.

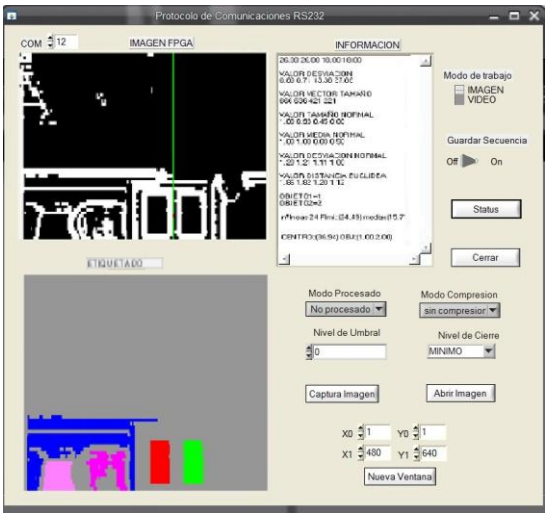

Fig. 8. Representación del cálculo de la línea central.

B. Procesado de imagen para la obtención de la variable de estado β

De la primera parte del procesamiento se obtienen las coordenadas del centro de la marca y la línea central que lo atraviesa. Así mismo también se tiene la información sobre las filas iniciales y finales de la zona central, el *hueco.* Esta información se utilizará por la segunda parte del procesamiento para el cálculo de β.

A la hora del cálculo se deben tener en cuenta dos supuestos:

- Si en la imagen no están los puntos finales de las líneas verticales principales: β se calcula a partir del punto medio y de la parte superior de la marca. Si las tres están a la misma distancia y en filas próximas, β será cero. No hay inclinación.

- Si en la imagen están los puntos finales de las líneas verticales principales: si, además, la posición de la fila perteneciente a la primera línea es mayor que la posición de la fila perteneciente a la segunda línea, β será positiva. El vehículo está situado a la izquierda de la marca. Si, por el contrario, la fila de la segunda línea es mayor, β será negativa. El vehículo está situado a la derecha de la marca.

Obteniendo las columnas de los cuatro puntos que definen el borde interno de la marca, calcular las columnas centrales no es complicado. Con los dos puntos laterales hallaremos β mediante la fórmula:

$$\beta = \tan^{-1}\left(\frac{f_{ci} - f_{cd}}{c_{cd} - c_{ci}}\right)$$

Donde:

f_{ci}: fila del punto a la izquierda del centro.
f_{cd}: fila del punto a la derecha del centro.
c_{ci}: columna del punto a la izquierda del centro.
c_{cd}: columna del punto a la derecha del centro.

V. CONCLUSIONES Y TRABAJOS FUTUROS

En este trabajo se propone la utilización de las FPGAs para la implementación de un sensor de visión de bajo coste con aplicaciones en robótica móvil.

Se ha desarrollado un prototipo que integra una parte importante del procesamiento de imagen para la detección de marcas de navegación. El sensor consigue una velocidad de procesamiento suficiente, limitada únicamente por el canal de comunicaciones, para aplicar técnicas de Aprendizaje por Refuerzo en el sistema de navegación del robot. También reduce notablemente la el ancho de banda en la transferencia de datos ente los subsistemas y no necesita de memoria externa.

La flexibilidad de las FPGAs permite incorporar módulos adicionales que facilitan la depuración del diseño durante la fase de desarrollo y durante la integración con el sistema de navegación.

Como líneas de trabajo futuro se plantean, en primer lugar, la incorporación en la FPGA del proceso completo de identificación de las marcas. Para ello será necesario integrar un softcore que se ocupe de las etapas postprocesado: etiquetado, identificación de los lóbulos y cálculo de las variables de estado. En segundo lugar se pretende trabajar en la identificación códigos matriciales tipo QR, que proporcionen una mayor información al subsistema de navegación.

VI. AGRADECIMIENTOS

Este trabajo ha sido financiado por el proyecto GV07/214 de la Generalitat Valenciana.

VII. REFERENCIAS

[1] R. Sutton and A. Barto, Reinforcement Learning, An Introduction, MIT Press, 1998.
[2] M.Y.I. Idris et al. Review of Feature Detection Techniques for SLAM and System on Chip Approach. Information Technology Journal 8 (3), 250-262, 2009
[3] Fowers, S. G. et. al. Vision aided stabilization and the development of a quad-rotor micro UAV. Inernational Symposium on Computational Intelligence in Robotics and Automation, Jacksonville, 2007.
[4] Chati, H.D. et. al. Hardware/Software co-desing of a key point detector on FPGA. International Symposium on Field-Programmable Custom Computing Machines 2007, Napa, CA.
[5] Beau Tippetts, Spencer Fowers, Kirt Lillywhite, Dah-Jye Lee and James Archibald. FPGA Implementation of a Feature Detection and Tracking Algorithm for Real-time Applications ISVC 2007, Part I, LNCS 4841, pp. 682–691, 2007.
[6] S. Jutchinson, G. Hager and P. Corke, A Tutorial on Visual Servo Control, IEEE Trans. on Robotics and Automation, 1996.
[7] D. Kragic and H.I. Christensen, Survey on Visual Servoing for Manipulation, Tech. Rep. ISRN KTH/NA/P--02/01--SE, CVAP259, 2002.
[8] T. Martinez-Marin and T. Duckett, Learning Visual Docking for Non-Holonomic Autonomous Vehicles, IEEE Intelligent Vehicles Symposium (IV'08), 2008.
[9] T. Martinez-Marin and T. Duckett, Learning Visual Docking for Non-Holonomic Autonomous Vehicles, IEEE Intelligent Vehicles Symposium (IV'08), 2008.
[10] http://www.mentor.com/products/fpga/handel-c/
[11] http://www.mentor.com/products/fpga/handel-c/pixelstreams/
[12] http://www.xilinx.com/
[13] *LabWindows* ®/CVI *User Interface Reference Manual*. National Instruments Corporation, Austin, TX (U.S.A.), 1996.