MAII - UPV

SISTEMA DE VISIÓN ACTIVO EMPOTRADO PARA ROBOT HUMANOIDE

Tesina de Máster

Autor:
PAU MUÑOZ BENAVENT

Directores:
JUAN FRANCISCO BLANES NOGUERA
ALFONS CRESPO LORENTE

Valencia, diciembre 2010

CONTENIDO

1. INTRODUCCIÓN Y MOTIVACIÓN

Los primeros experimentos realizados en el campo de la visión por computador datan de finales de 1950 y la gran mayoría de los conceptos que hoy se consideran esenciales fueron introducidos en los años 80. Desde entonces, la visión por computador representa uno de los campos de la inteligencia artificial que más ha crecido en los últimos años. El objetivo de todo sistema de visión por computador es la interpretación de una escena a través de las imágenes capturadas por el sensor de visión (transformación imagen-información), más allá del procesamiento de imágenes (transformación imagen-imagen).

La visión por computador es una materia con aplicaciones múltiples en campos tan diversos como medicina, biología, meteorología, seguridad, transporte, entretenimiento, domótica y robótica.

En el caso de la robótica, los sistemas de visión se están convirtiendo en la principal fuente de información sensorial. Uno de los desarrollos que más auge está tomando hoy en día son los robots humanoides. Por un lado plantean retos a resolver (estabilidad, sensorización, integración de sistemas de visión) de gran interés desde el punto de vista de los sistemas empotrados de control, y por otro lado su natural orientación a la imitación del ser humano, ofrece un campo de trabajo en cuestiones como la inteligencia artificial, la interacción hombre-máquina y la cooperación entre robots como ninguna otra plataforma robótica ofrece.

1.1. OBJETIVOS

El principal objetivo de este trabajo consiste en desarrollar un sistema de visión activo e integrarlo en la arquitectura de control distribuida de un robot humanoide como sensor inteligente, capaz de interpretar el entorno y generar la información a compartir con el resto del sistema de control.

El trabajo se ha realizado sobre el robot humanoide MicroBiro-II del Instituto de Automática e Informática Industrial de la Universidad Politécnica de Valencia (1), (2), (3)

Los objetivos particulares en que se ha organizado la labor de investigación son:

- Estudio detallado de las arquitecturas de control desarrolladas por distintos grupos de investigación en robótica humanoide, participantes en las competiciones Robocup.
- Desarrollo de un sistema de visión activo empotrado.
    - Implementación multihilo sobre DSP.
    - Comunicación con los servomotores.
    - *Image-based visual servoing* para el *tracking* de objetos.
    - Estudio e implementación de algoritmos para interpretación de un entorno particular tipo Robocup.
- Integración del sistema de visión en la arquitectura de control del robot humanoide.
    - Canal de comunicaciones
    - Definición del protocolo de mensajes
- Desarrollo de una interfaz gráfica de control del sistema de visión.
    - Configuración de los parámetros de la cámara.
    - Calibración de los colores de los objetos.
    - Banco de pruebas de algoritmos de procesamiento de imagen.

1.2. ROBOCUP

Una de las formas más productivas de analizar el comportamiento de los diseños son las competiciones, ya que permiten comparar los desarrollos con los de otros grupos de investigación y extraer conclusiones. En esta línea se enmarcan los eventos RoboCup, que sirven de punto de encuentro de numerosos grupos de robótica humanoide de todo el mundo.

RoboCup™ (originalmente llamado *Robot World Cup Initiative*) es una iniciativa internacional de investigación y educación. Es un intento de fomentar la inteligencia artificial y la investigación robótica inteligente, proporcionando un problema estándar donde una amplia gama de tecnologías pueden ser integradas y examinadas.

Con este fin, RoboCup eligió utilizar el juego del fútbol como uno de los dominios principales, y organizó la llamada R*oboCup Soccer League*. Para que un equipo de robots pueda realizar un partido de fútbol, varias tecnologías deben ser incorporadas, como principios de diseño de agentes autónomos, colaboración multi-agente, razonamiento en tiempo real, robótica y fusión sensorial. Entre las tareas a realizar: movimientos del robot (caminar, chutar…), interpretación del entorno, autolocalización y comunicación entre robots, entre otras.

A parte de la competición, se realizan otras actividades y debates durante la RoboCup: jornadas técnicas, programas de educación, desarrollo de la infraestructura, etc.

En la Liga Humanoide de la *RoboCup Soccer League* los robots deben tener una construcción similar al cuerpo humano y utilizar los sentidos de forma similar a los humanos. Esto conlleva restricciones en el diseño mecánico, en los sensores a utilizar (sólo sensores propioceptivos, excepto el sistema de visión) y en el diseño del sistema de visión (la limitación del ángulo de visión a 180º y del máximo de número de cámaras a utilizar a 2), entre otros. En función del tamaño del robot se diferencian tres ligas dentro de la Humanoide: *KidSize* (30-60 cm), *TeenSize* (100-120 cm) y *AdultSize* (más de 130 cm).

La *KidSize Humanoid League*, donde se pretende participar con el desarrollo aquí propuesto, consiste de una serie de pruebas técnicas y partidos entre robots enmarcados en el fútbol robótico. Para ayudar en el procesamiento del entorno, los objetos de interés (porterías, pelota, campo y líneas) se identifican con colores alejados en el espectro de colores, de forma que una posible primera diferenciación pueda realizarse por colores.

FIGURA 1. ENTORNO TIPO ROBOCUP

Si bien es cierto que como escenario de trabajo se plantea un entorno tipo RoboCup, la propuesta de hardware y arquitectura de control está pensada para poder ser aplicada en cualquier entorno cambiante y garantizar la caracterización de un entorno dinámico en tiempo real.

1.3. CONTENIDO DEL DOCUMENTO

Este documento se estructura de la siguiente forma:

- En el apartado 2 se hace una revisión del estado del arte de la visión por computador aplicada a robots humanoides, haciendo mención especial de los integrantes más destacados de la *KidSize Humanoid League 2010*.
- En el apartado 3 se presenta el trabajo realizado, que incluye:
  - La labor desarrollada con los robots humanoides comerciales Bioloid y Nao.
  - La integración del sistema de visión en la arquitectura de control distribuida del robot.
  - La implementación del sistema de visión del robot.
  - El desarrollo de una interfaz gráfica de interacción con el sistema de visión del robot.
  - Un ejemplo de aplicación del nuevo desarrollo del robot.
- Las conclusiones y las líneas de trabajo futuro se presentan en el punto 4.

2. ESTADO DEL ARTE

En este apartado se presenta el trabajo hecho en campos destacados de la visión por computador aplicada a los robots humanoides, *visual servoing* y planificación de trayectorias y evitación de obstáculos. Además, se hace un estudio a nivel de hardware de visión, arquitectura de control y procesamiento del entorno, de los robots desarrollados por los grupos de investigación participantes en la *Kid Size Humanoid League* de la Robocup 2010.

2.1. VISUAL SERVOING

Se define el término *visual servoing* como la capacidad que proporciona la visión artificial de definir bucles de control en lazo cerrado para el control de la posición de un robot. En anterioridad a la utilización de este término, el menos específico *visual feedback* era utilizado generalmente.

La función del *visual servoing* es utilizar la información visual para controlar la posición de un robot, ya sea la posición del efector final en robots manipuladores o bien la posición del robot en robots móviles, respecto de objetos o marcas.

Los primeros sistemas sobre *visual servoing* fueron introducidos en los años 1980s. Desde entonces el número de publicaciones de investigación en este ámbito ha ido creciendo, especialmente en los últimos años, auspiciado por la capacidad de los computadores, que propiciaron superar la barrera del análisis de escenas a velocidad suficiente como para servocontrolar un robot.

Las aplicaciones propuestas abarcan temas tan diversos como procesos de fabricación, teleoperación, *tracking* de objetos, robótica de entretenimiento, control de la dirección de coches, aterrizaje de aviones, etc.

Visual servoing supone la fusión de resultados de muchas áreas más elementales, incluyendo procesado de imágenes, cinemática, dinámica, teoría de control, computación en tiempo real y visión activa. El objetivo del visual servoing es el control del robot para modificar su entorno usando la visión, más allá de la simple observación del entorno.

En 1980, Sanderson and Weiss (*Image-based visual servo control using relational graph error signals.* 1980) presentaron una taxonomía de los sistemas de *visual servoing*, en la que todos los sistemas posteriores pueden ser clasificados. Se basaba en dos cuestiones esenciales:

- ¿La estructura de control es jerárquica, con el sistema de visión proporcionando referencias al control de articulaciones, o es directamente el controlador del sistema de visión el que controla las articulaciones?
- ¿La señal de error de posición se define en coordinadas 3D (espacio de trabajo) o directamente en términos de la imagen?

Si la estructura de control es jerárquica y usa el sistema de visión para proporcionar referencias al controlador de articulaciones, se habla de sistemas *dyanmic look-and-move*. Por el contrario, se habla de *direct visual servo* si se elimina el controlador de articulaciones y se reemplaza por un controlador que directamente calcula la posición de las articulaciones.

La segunda clasificación, distingue entre control basado en posición (*position-based*) y control basado en la imagen (*image-based*). En los sistemas *position-based*, las características son extraídas de la imagen y utilizadas en conjunción con un modelo de la cámara para estimar su posición respecto de la cámara. En

cambio, en los sitemas *image-based*, las acciones de control son calculadas directamente a partir de las características extraídas de la imagen.

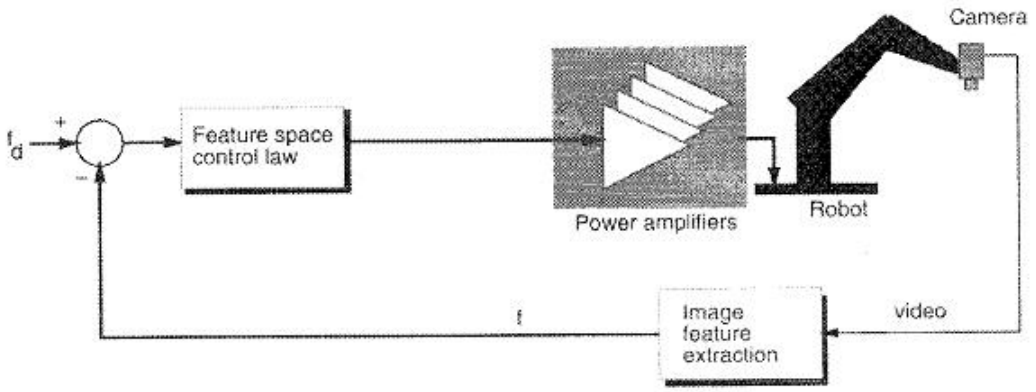


FIGURA 2. IMAGE-BASED VISUAL SERVOING.

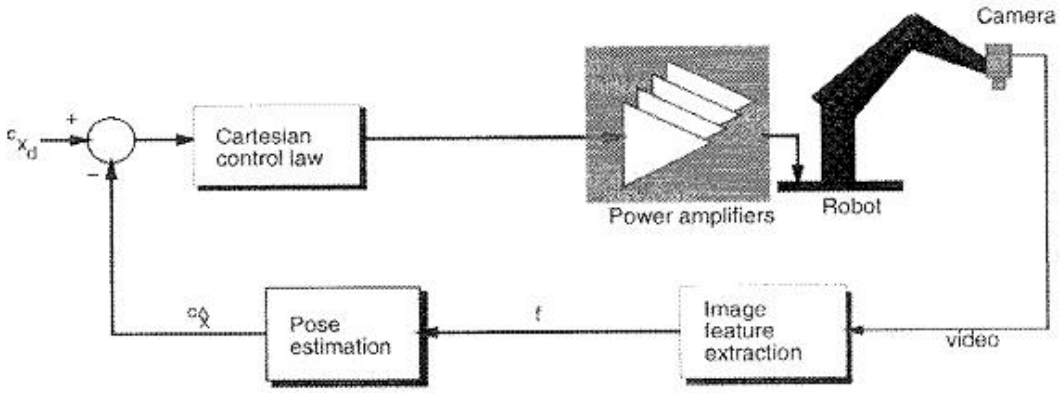


FIGURA 3.POSITION-BASED VISUAL SERVOING

Tal y como se verá en el capítulo correspondiente, ambas técnicas de *visual servoing* se aplican en este trabajo: *image-based visual servoing* para el *tracking* de objetos y *position-based visual servoing* para el movimiento del robot en el campo.

2.1.1. Visual Servoing for Dual Arm Motions on a Humanoid Robot

En este trabajo (4) se presenta un sistema de *visual servoing* que permite a un robot humanoide ejecutar robustamente tareas de manipulación y agarre con ambas manos. Los objetos y ambas manos son seguidos alternativamente en la imagen y un controlador combinado bucle abierto – bucle cerrado se utiliza para posicionar las manos respecto del objeto. Además, se presenta un marco de control para posicionamiento reactivo de ambas manos usando *position-based visual servoing*, donde los datos provenientes del sistema de visión, los encóders de las articulaciones y los sensores de par y fuerza son fusionados y los valores de velocidad generados. Este marco ha sido probado para manipulaciones y agarre con ambas manos con el robot Armar-III en un entorno de cocina.

Las imágenes provenientes del sistema de visión son procesadas con algoritmos de reconocimiento y localización. Además, la posición de los objetos y la posición del efector final son determinados en el espacio cartesiano. El control visual ofrece la posibilidad de supervisar la ejecución de las tareas de agarre o manipulación sin la necesidad de calibrar el sistema *mano-ojo*.

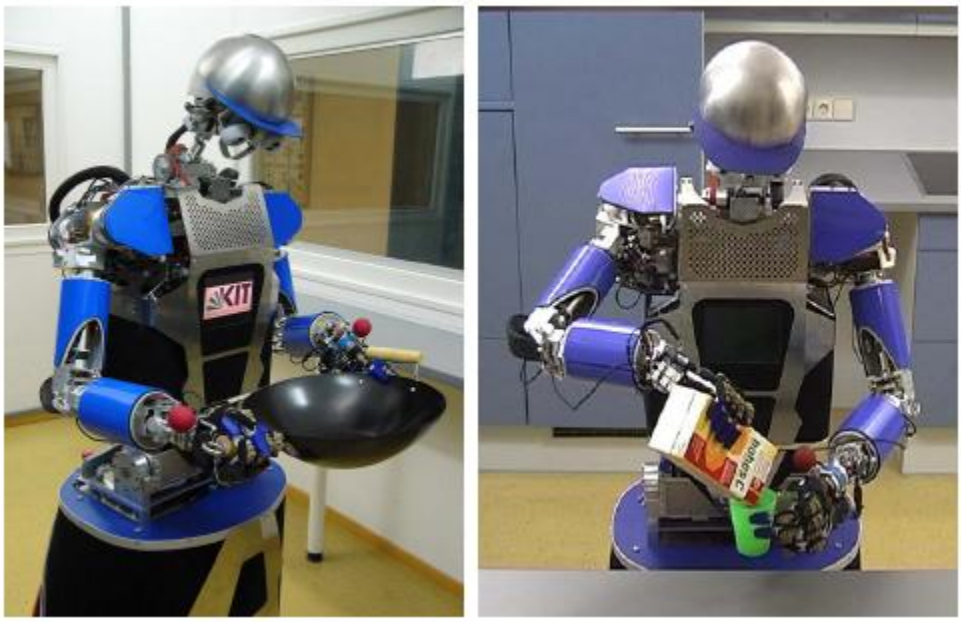


FIGURA 4. EL ROBOT ARMAR-III REALIZANDO TAREAS DE AGARRE Y MANIPULACIÓN

2.1.2. Visually-Guided Grasping while Walking on a Humanoid Robot

La principal idea aportada en el artículo (5) es la división del control en distintas tareas de control, en función de los sensores presentes, que son ejecutadas simultáneamente. Esta estructura es aplicada para una tarea de *visual servoing*, en concreto, el robot camina por una trayectoria previamente planificada, manteniendo el objeto en su campo de visión y, finalmente, cuando está suficientemente cerca, el robot coge el objeto mientras sigue caminando.

Una buena estimación de la posición del objeto se obtiene a partir del sistema de visión estéreo montado en la cabeza del robot. La primera tarea es mantener el objeto centrado en la imagen a través del *visual servoing* durante el movimiento del robot, para asegurar su visibilidad. La segunda tarea mueve la garra del robot a la posición del objeto para poderla coger. Finalmente, se tienen en cuenta las restricciones de las articulaciones. Los experimentos se realizaron sobre el robot HRP-2.

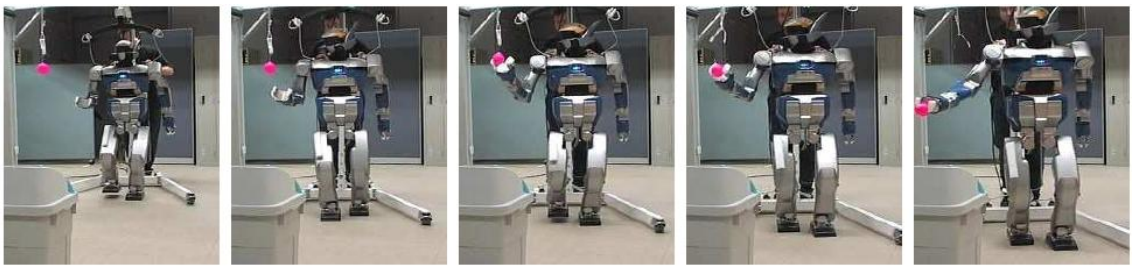


FIGURA 5. EL ROBOT HRP-2 COGIENDO UN OBJETO MIENTRAS CAMINA

2.2. PLANIFICACIÓN DE TRAYECTORIAS Y EVITACIÓN DE OBSTÁCULOS.

2.2.1. Obstacle avoidance and path planning for humanoid robots using stereo vision

Este artículo (6) presenta métodos para planificación de trayectorias y evitación de obstáculos para el robot humanoide QRIO (Figura 6), permitiéndole caminar de forma autónoma en un entorno doméstico. La estrategia para la evitación de obstáculos se basa en la estimación del plano del suelo a partir de los datos capturados por el sistema de visión estéreo.

FIGURA 6. EL ROBOT QRIO DE SONY

El hardware del sistema de visión del robot QRIO consiste en dos cameras CCD en color (con resolución 352x288 píxeles), un módulo FPGA para el procesamiento estéreo, dos unidades SDRAM de 8Mbyte, una flash ROM y un microprocesador de 8 bits, tal y cómo muestra la Figura 7. El sistema de visión recibe un par de imágenes de las cámaras y procesa la disparidad entre ellas usando *block matching*. La imagen de disparidad resultante se envía a la unidad principal como una señal de vídeo YUV. Los parámetros de control de la cámara, cómo AGC, AWB y parámetros de control estéreo se pueden definir a través de un puerto serie especial (OPEN-R Bus) entre el microprocesador y la CPU principal.

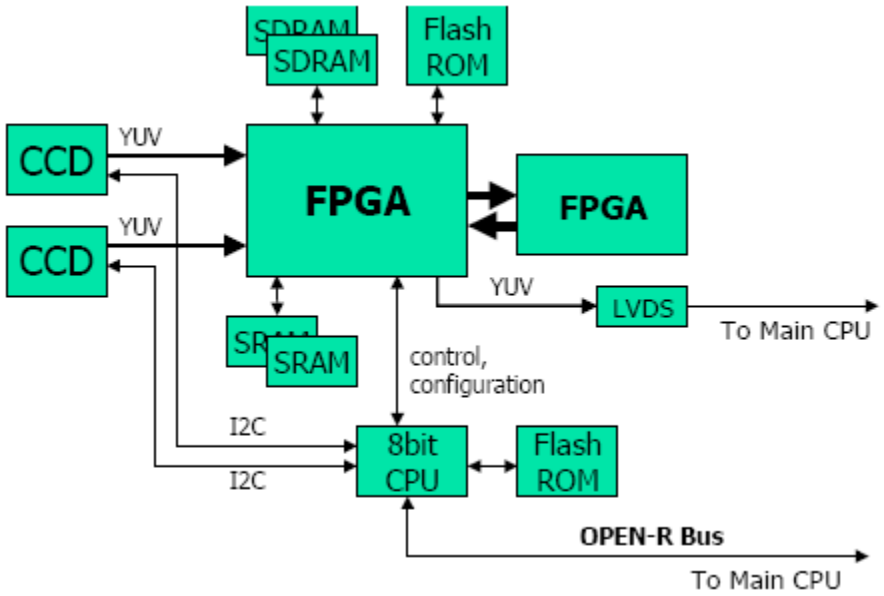


FIGURA 7. HARDWARE DEL SISTEMA DE VISIÓN DE QRIO

2.2.2. Stair climbing for humanoid robots using stereo vision

En este trabajo (7), los mismos autores del artículo anterior utilizan el robot humanoide QRIO para subir y bajar escaleras, a partir de los datos capturados por el sistema de visión estéreo explicado anteriormente y procesados para detectar los distintos planos. El sistema se presenta en la Figura 8.

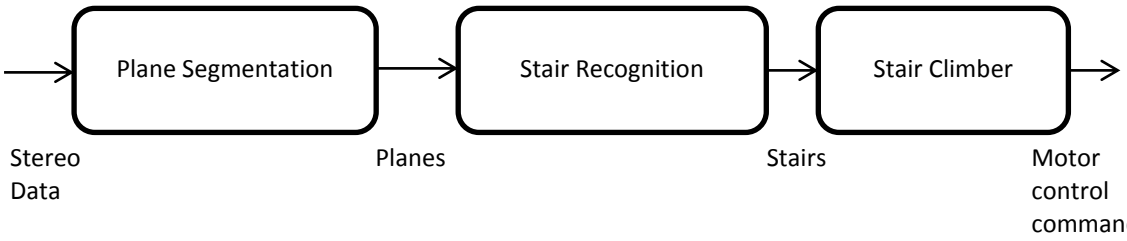


FIGURA 8. SISTEMA PARA RECONOCER Y SUBIR ESCALONES

2.2.3. Vision-Guided Humanoid Footstep Planning for Dynamic Environments

En este artículo (8) se presenta un diseño para dotar a un robot humanoide de la capacidad de caminar de forma autónoma, permitiéndole dirigirse a la posición deseada evitando los obstáculos. Un mapa del entorno, incluyendo el robot, el objetivo y la localización de los obstáculos se construye en tiempo real a partir de la información del sistema de visión estéreo. Después, un planificador de pasos calcula la secuencia óptima dentro de un horizonte temporal. Estos cálculos se rehacen, total o parcialmente, en el caso de que el entorno cambie. Los experimentos se han realizado sobre un robot ASIMO de Honda, con el sistema de visión modificado.

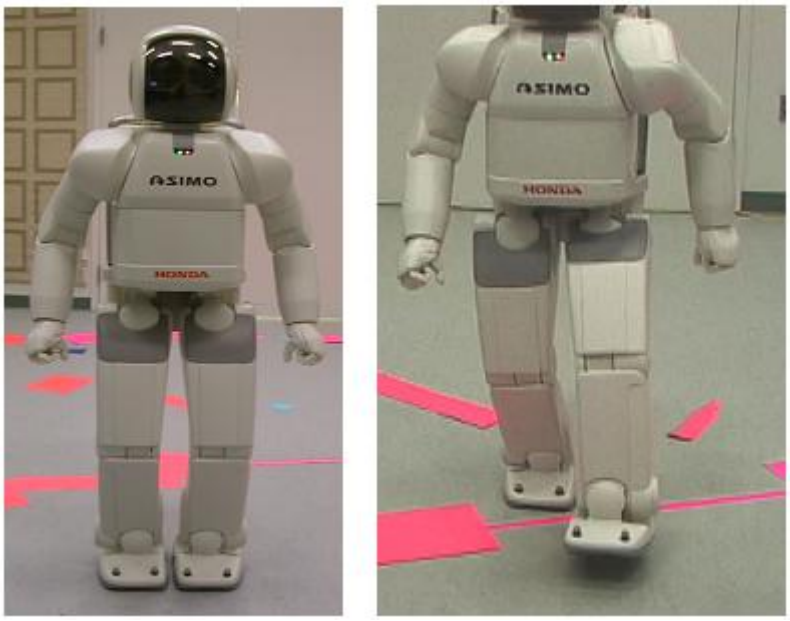


FIGURA 9. EL ROBOT ASIMO DE HONDA

En realidad, el sistema de visión utilizado para estos experimentos consta de una cámara cenital, en concreto una cámara industrial montada a 3.5 m del suelo, abarcando un área de visión de 3.2 m x 2.4 m. La cámara proporciona imágenes de 640 x 480 píxeles a 25-30 fps. La Figura 10 muestra el sistema de sensorización utilizado.

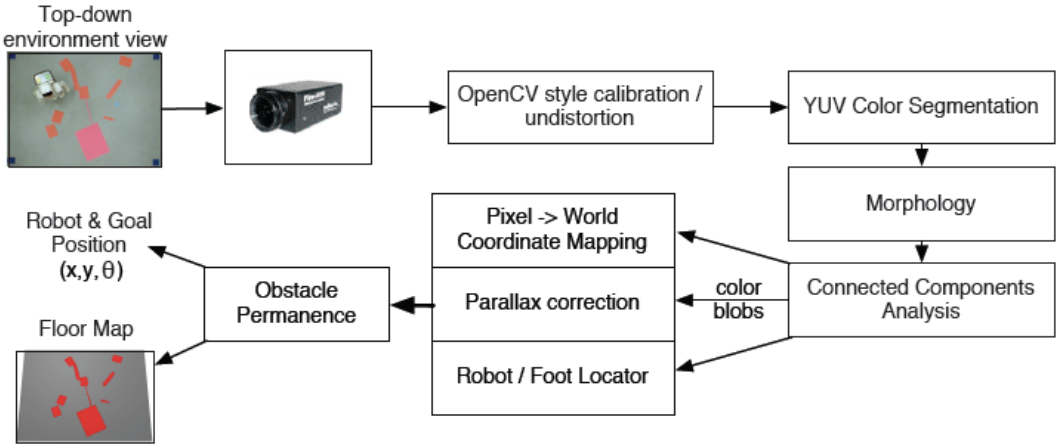


FIGURA 10. VISTA GENERAL DEL PROCESO DE SENSORIZACIÓN DESARROLLADO PARA EL ROBOT ASIMO

La cámara es calibrada offline, utilizando OpenCV para calcular los parámetros intrínsecos y extrínsecos así como también los coeficientes radiales de distorsión. Se utilizan marcas de color para localizar los objetos en el suelo, el objetivo, la posición actual del robot y los límites del área dónde se puede mover el robot. La segmentación por color se realizada directamente sobre la imagen en YUV generada por la cámara. Se definen offline los umbrales de color para segmentar los obstáculos u otras marcas y se aplican algoritmos de conectividad de píxeles del mismo color, que proporcionan información sobre los objetos (centroide, área, ejes mayor/menor, orientación). Posteriormente, se hace una transformación a coordenadas globales.

2.2.4. Locomotion planning of humanoid robots to pass through narrow spaces.

El método propuesto en este artículo (9) genera un mapa local del entorno en 3D a partir de la información visual y escoge la locomoción adecuada entre caminar bípedo (con altura y anchura de paso) y gatear.

Utilizan el robot humanoide HRP-2 mostrado en la Figura 11. HRP-2 posee un sistema de visión estéreo compuesto por tres cámaras. Dos cámaras horizontales están separadas por 144 mm, y la tercera cámara está 70 mm por encima de estas. Se adopta el sistema de tres cámaras por la dificultad que conlleva, en un sistema de dos cámaras, detectar la línea horizonte. La velocidad del obturador es controlable por un computador para adaptarse a condiciones de iluminación variables.

El procesamiento de la imagen está basado en el sistema VVV [*R&D of Versatile 3D Vision System, 1998, Tomita et al.*]. VVV está compuesto por diversos módulos de procesamiento de imágenes, reconstrucción de entornos 3D, reconocimiento de objetos y *tracking* de objetos, entre otros.

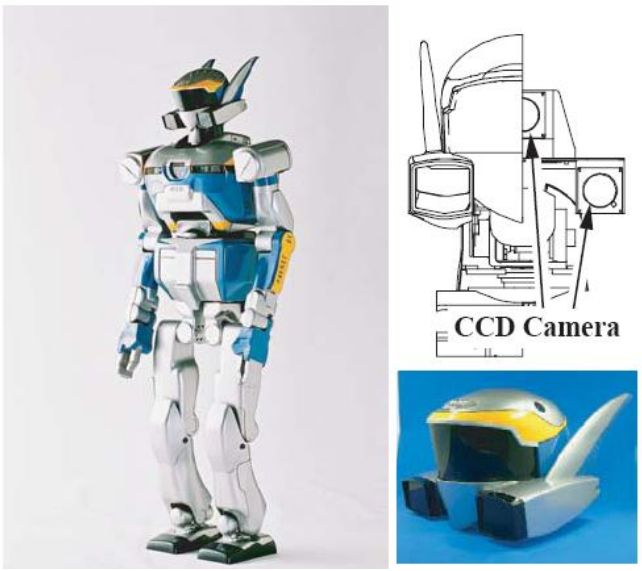


FIGURA 11. EL ROBOT HRP-2

2.3. ROBOTS HUMANOIDES EN LA ROBOCUP 2010

Desde el año 1997 en que se creó la Robocup, la evolución de los robots humanoides y, en concreto, de los sistemas de visión y su integración en las arquitecturas de control, ha sido constante. Se presentan a continuación las distintas propuestas de hardware para el sistema de visión y de integración en la arquitectura de control, así como las técnicas para el procesamiento del entorno de los equipos participantes en la liga *Humanoid KidSize League* de la edición RoboCup 2010. Además, se entra en detalle en alguno de los equipos más destacados de la edición.

Las distintas propuestas de hardware para el sistema de visión pueden agruparse en:

- Webcams conectadas a una PDA dedicada de propósito general: (10) y (11).
- Cámaras conectadas a la placa principal de procesamiento (caso de webcams: (12), (13), (14), (15) (16) (17) (18); cámaras CMOS: (19), (20), (21), (22), (23), (24); o cámaras CCD: (25)).

- Cámaras CMOS con microcontrolador dedicado: (26), (27). La gran mayoría de los grupos presentan propuestas de visión monoscópica (una única cámara), a excepción de (23) y (24), que presentan un sistema de visión estéreo o estereoscópica (dos cámaras).

En cuanto al procesamiento del entorno, los pasos básicos sobre los que trabajan todos los grupos se muestra en la Figura 12.

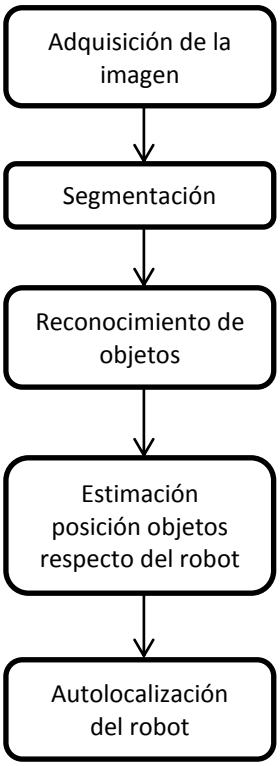


FIGURA 12. PASOS BÁSICOS EN EL PROCESAMIENTO DEL ENTORNO

El hecho diferenciador entre grupos es la implementación de cada uno de los pasos.

- Distintas resoluciones de la imagen y distintos espacios de color.
- Segmentación por *image scoring* en lugar del típico por umbrales de color (16).
- Adición de algunas funcionalidades extra como detección de líneas (15) y (24), planificación de trayectorias de los robots a partir de la caracterización del entorno (23), identificación de robots (11) e intercambio de mensajes entre robots para coordinación y fusión de información (17), entre otras.

2.3.1. Darmstadt Dribblers

Los robots están equipados con cámaras articuladas y hardware para computación distribuido, consistente en un controlador para generación de la locomoción y control de estabilidad y un PC empotrado para todas las demás funciones.

El poder computacional del sistema para el procesamiento de la información se divide en tres niveles. El nivel más bajo de computación se realiza en cada uno de los 21 servomotores. Cada servomotor está equipado con un microcontrolador para control de posición y velocidad con parámetros ajustables. Tareas más estrictas de tiempo real como la generación de movimientos y control de estabilidad se

ejecutan en un microcontrolador (nivel de reflejos). Control de alto nivel como la visión, el modelado del entorno, el control de comportamientos y coordinación del equipo se ejecutan en un PC embebido estándar (nivel cognitivo). Los tres niveles del software de control se comunican a través de una conexión serie.

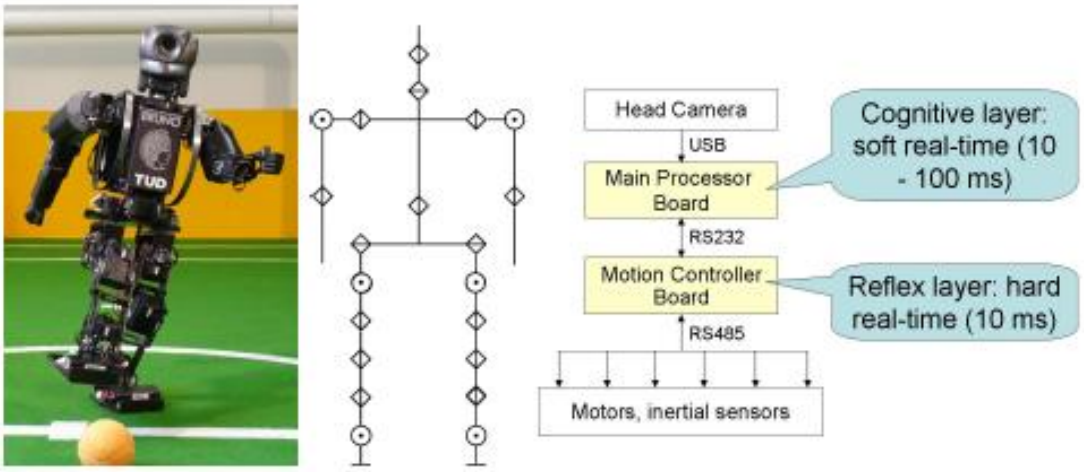


FIGURA 13. ROBOT HUMANOIDE DE DARMSTADT DRIBBLERS

Procesamiento de la imagen. Está dividido en dos partes: una etapa de preprocesamiento y múltiples módulos para el reconocimiento de objetos. Dicho reconocimiento, utilizando los denominados perceptores, pueden trabajar con múltiples tipos de imagen, como por ejemplo imágenes segmentadas, imágenes en escala de grises o la imagen cruda directamente. Así, dependiendo del objeto y del algoritmo de reconocimiento, el nivel de abstracción adecuado puede ser utilizado por cada perceptor, manteniendo al mínimo los esfuerzos de preprocesamiento. Los perceptores desarrollados hasta el momento detectan líneas del campo, cruces de líneas, el círculo del centro del campo, la pelota, las porterías, las guías y los obstáculos.

El modelado del entorno consiste en un conjunto de modelos que son actualizados usando la información extraída por el módulo de visión. Una parte del modelado del entorno es la autolocalización, que se consigue utilizando localización Markov con filtro de partículas.

2.3.2. FUmanoid

Cada robot posee un empotrado Gumstix Verdex PRO XL6P. El procesador es un Marvell PXA270 con XScale a 600 MHz. Dicho empotrado tiene distintas características que lo hacen ideal para usar como cerebro de robots humanoides. Entre las características, peso reducido y bajo consumo, conexión directa a cámara y facilidad de extensión. Todas las unidades hardware incluyendo motores y sensores (entre ellos la cámara) se conectan directamente al procesador principal a través del bus RS485.

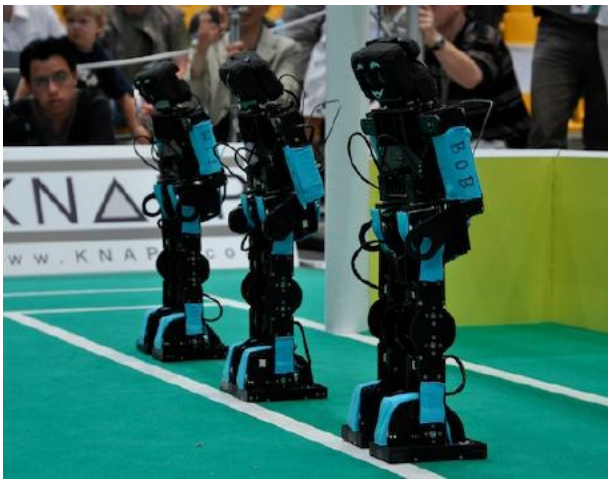


FIGURA 14. ROBOTS FUMANOID

El diseño software se estructura en bloques. Los principales bloques son: interfaz hardware, visión y localización, planificación y red.

Interfaz hardware. Contiene las rutinas de bajo nivel para el acceso al hardware del robot, incluyendo sensores y actuadores.

Visión y localización. Manejo de la cámara y algoritmos de procesamiento de imágenes, como reconocimiento de guías y otros objetos. La autolocalización se realiza utilizando filtro de partículas, con la información proveniente del sistema de visión y de la odometría.

Planificación. Organizado en una estructura multinivel y multihilo. Los niveles se denominan Estrategia, Rol, Comportamiento y Movimiento. Cada nivel contiene un Escenario que corre en paralelo con los escenarios de los otros niveles. Un escenario de más alto nivel puede terminar y alterar el escenario de un nivel inferior, sin embargo se suele hacer en sincronización para evitar conflictos e inestabilidades.

Red. Responsable de las comunicaciones inalámbricas de cada robot con el de otros robots, vía WLAN.

2.3.3. CIT Brains Kid

El robot tiene dos placas de control: una para locomoción y otra para reconocimiento de imagen, determinación de comportamientos a realizar y otras cosas. Esta última CPU es ligera y potente, con sistema operativo NetBSD, con lo que se pueden utilizar herramientas de UNIX. Además, contiene una FPGA, que ayuda a la computación en tiempo real.

La cámara envía la señal de la imagen a la CPU principal, que la guarda en un *buffer* de memoria. La CPU procesa los datos de la imagen para detectar la posición de la pelota, los robots y las guías. A partir de la posición de dichas guías, el robot estima su posición utilizando filtro de partículas. A partir de esta información, el robot decide el siguiente comportamiento a realizar.

El comando de la acción se envía a la segunda CPU por RS232, que lo descodifica y ejecuta la acción.

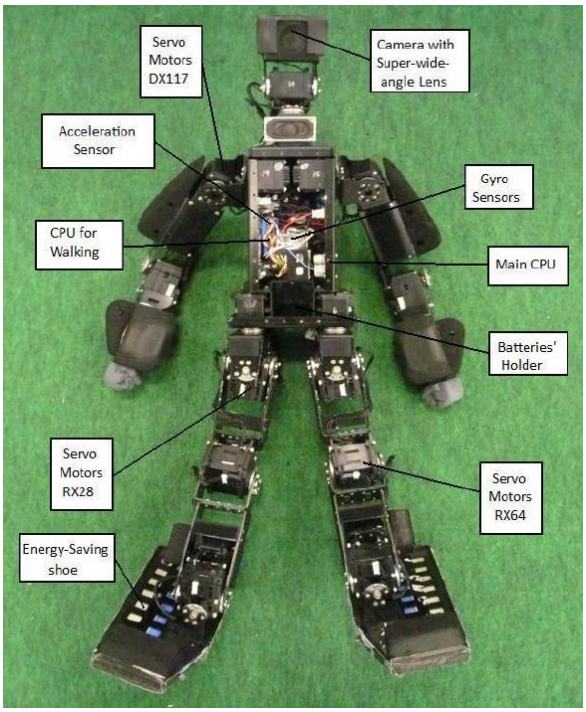


FIGURA 15. ROBOT CIT BRAINS KID

Image processing and position estimation. La CPU procesa imágenes de resolución 160x120 a 30 fps. Se ha desarrollado una interfaz gráfica para calibrar los umbrales de color, necesarios para la segmentación de la imagen. Se estima la posición del robot respecto las guías del campo utilizando filtro de partículas, sin embargo la precisión de la estimación es insuficiente, por lo que se está desarrollando un algoritmo para detectar las líneas del campo.

2.3.4. Team DARwIn

La serie de robots DARwIn (*Dynamic Anthropomorphic Robot with Intelligence*) es una familia de robots humanoides desarrollados por la Universidad de Virginia Tech y la Universidad de Pennsylvania.

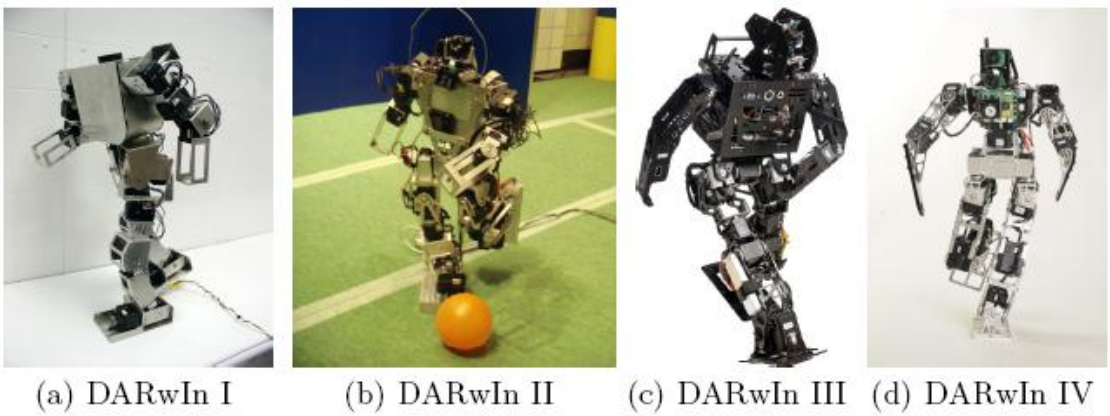


FIGURA 16. LA FAMILIA DE ROBOTS DARWIN

Las tareas de computación se llevan cabo en el Compulabs Fit-PC2, que posee un Intel Atom Z530, con un 1GB de RAM y WiFi incorporada. El computador está conectado a una cámara Philips SPC1300 y a un microcontrolador Arduino,

La placa Arduino, que integra un ATmega1280, actúa como puente de comunicación con los motores Dynamixel, además de adquirir y procesar las señales de los sensores. El diagrama de bloques se muestra en la Figura 17.

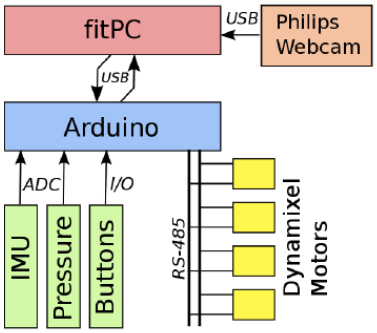


FIGURA 17. ARQUITECTURA HARDWARE DE LA FAMILIA DE ROBOTS DARWIN

La arquitectura es novedosa, puesto que utiliza MATLAB como plataforma de desarrollo. Las interfaces a nivel hardware están implementadas como rutinas compiladas Mex, instanciables desde Matlab. Proveen acceso a la cámara y al microcontrolador que, a su vez, se comunica con los servos y la IMU (*Inertial Measurement Unit*).

Vision. La calibración de los colores se realiza utilizando una herramienta MATLAB. Una vez segmentada la imagen utilizando *look-up tables*, se analizan los segmentos con distintos atributos, como tamaño, forma y posición, para poderlos clasificar. Se realiza la autolocalización del robot a partir de la información de los postes de la portería.

2.3.5. aiRobots

El hardware de aiRobot-III se puede dividir en cuatro módulos: estrategia y control de movimientos, actuadores, sistema de visión y multisensorial.

Estrategia y control de movimientos. Se utiliza una placa de evaluación NIOSII Cyclone II de Altera. Después de generar el sistema utilizando SOPC Builder (parte del software Quartus II de Altera), se sintetizan los circuitos de motores y sensores con puertas lógicas FPGA, utilizando Verilog HDL (*Hardware Description Language*). Finalmente, se desarrollan las aplicaciones en C/C++ para distintos componentes del sistema del sistema de control, como controlador borroso y estrategia inteligente.

Actuadores. Se utilizan productos de Dynamixel de Robotis, que permiten acceder a variables internas, como posición, velocidad y señal de realimentación.

Sistema de visión. Se puede dividir en dos partes: el dispositivo de captura de imágenes y el centro de procesamiento de imágenes. Se utiliza una webcam Logitech QuickCam Pro5000 como dispositivo de captura y una PDA ACER N300, como centro de procesamiento.

FIGURA 18. SISTEMA DE VISIÓN DEL ROBOT AIROBOT-III

Módulo multisensorial. Para aumentar la robustez y la estabilidad del robot, y hacer frente a las perturbaciones externas, se utiliza un acelerómetro y una brújula digital.

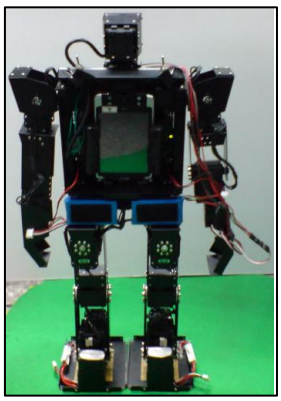


FIGURA 19. EL ROBOT AIROBOT-III

2.3.6. CONCLUSIONES

Se observa una tendencia clara a la utilización de sistemas distribuidos para incorporar una tarea pesada como es el procesamiento del entorno. Sin embargo, se considera novedosa nuestra propuesta de utilización de un procesador de señales (DSP).

3. TRABAJO REALIZADO

Con el fin de resaltar nuestra propuesta de arquitectura de control distribuida con el sistema de visión activo, se analizará previamente la arquitectura de control de los robots Nao V3 y V3+ de Aldebaran Robotics®, utilizados en la *Standard Platform League* de la Robocup. Además, se presentará el trabajo realizado con los robots Bioloid de Robotis® y el módulo de visión empotrado HaViMo, así como las pruebas y resultados obtenidos en los mismos.

3.1. TRABAJOS PREVIOS – BIOLOID

3.1.1. Descripción

Bioloid es un kit robótico desarrollado por la empresa coreana Robotis® que permite construir diferentes tipos de robots a partir de los componentes disponibles: servoactuadores, sensores, uniones y CPU.

Una de las configuraciones posibles, y con la que se ha trabajado, es el robot humanoide. Esta construcción consta de 18 grados de libertad (18 servoactuadores AX-12+ conectados por bus serie TTL) más la CPU con el procesador ATMEL ATMega128 @ 16 MHz.

FIGURA 20. ROBOT BIOLOID HUMANOIDE DE ROBOTIS

De entre las aportaciones hechas para el robot Bioloid, cabe destacar el desarrollo de un módulo empotrado de visión (HaViMo) presentado por la universidad FU-Berlin.

Dicho módulo se presenta encapsulado con la carcasa Dynamixel y accesible por bus serie TTL. Es capaz de procesar imágenes a 8 fps a una resolución QQVGA (160x120) y caracterizar hasta 15 agrupaciones de pixeles de colores en la imagen.

Se proporciona una herramienta software de calibración para construir una *look-up table* con las agrupaciones de colores a caracterizar en la imagen y para modificar los registros de la cámara.

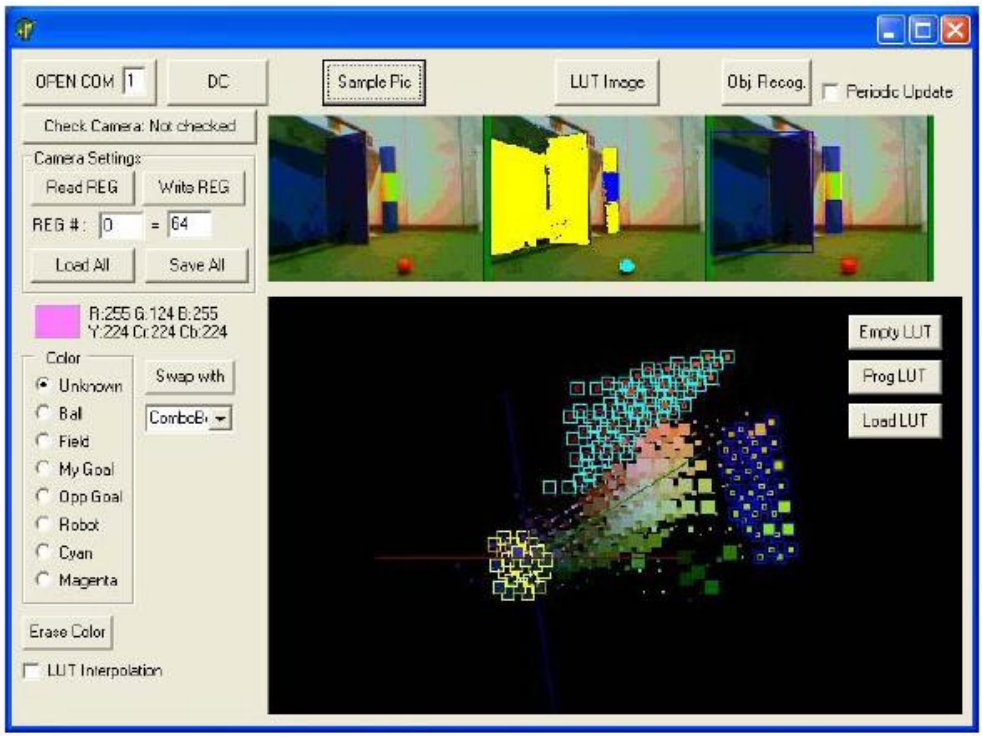


FIGURA 21. SOFTWARE DE CALIBRACIÓN HAVIMO

El módulo proporciona la información ordenada en una matriz:

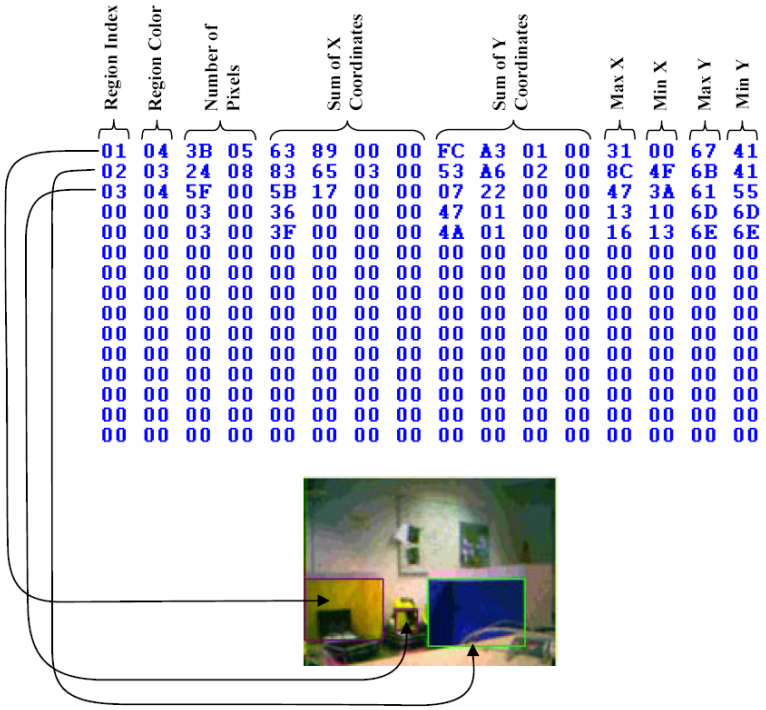


FIGURA 22. INFORMACIÓN PROPORCIONADA POR EL MÓDULO HAVIMO

3.1.2. Resultados

Con el módulo de visión HaViMo en el Bioloid se realizó la implementación de un portero de fútbol. El procesamiento de los datos entregados por el módulo permitía hacer *tracking* a la pelota y calcular distancias, de forma que el robot decidía tirarse hacía el lado donde se dirigía la pelota cuando estaba suficientemente cerca.

El trabajo realizado con el módulo de visión ha sido fundamental para entender el procedimiento utilizado por este grupo de investigación en la interpretación del entorno (similar al realizado por todos los grupos participantes en la Robocup) y ha proporcionado algunas ideas para la realización de nuestra propuesta.

3.1.3. Limitaciones

- Baja capacidad de procesamiento, limitada a un conjunto de objetos predefinidos.
- Los algoritmos de procesamiento de la imagen no son modificables.
- La información necesita un post-procesamiento para identificar los objetos de interés y calcular su posición respecto del robot.
- No es un sistema de visión activo: el sistema de visión no es capaz de interpretar el entorno y decidir la siguiente acción a realizar, sino que necesita para ello a la unidad de procesamiento.

3.2. OTRAS ARQUITECTURAS – NAO

3.2.1. Descripción

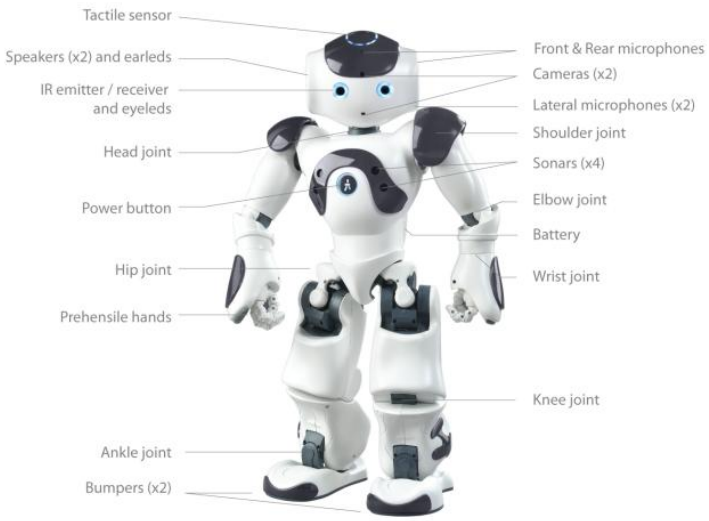


FIGURA 23. ROBOT NAO DE ALDEBARAN ROBOTICS

Los Nao son los robots utilizados por todos los equipos en la *Standard Platform League* de la Robocup. Sus características se muestran en la Tabla 1.

CONSTRUCCIÓN	58 cm y 4.8 kg 23 grados de libertad
ENERGÍA	Batería Li-Ion, 6 celdas en serie, Un=21.6V, capacidad: 2Ah
ACTUADORES	Motores DC Coreless MAXON Microcontroladores dsPIC
UNIDAD DE CONTROL PRINCIPAL	x86 AMD GEODE 500MHz CPU 256 MB SDRAM / 2 GB memoria Flash
SOPORTE DE EJECUCIÓN	SO Embedded Linux
PERIFÉRICOS	1 conectores USB (WiFi) 1 conector Ethernet
CONJUNTOS DE SENSORES	Giróscopos Acelerómetro *Bumpers* Sónar I/R Cámara CMOS

TABLA 1. CARACTERÍSTICAS GENERALES NAO

La arquitectura desarrollada para el robot Nao (28) se ha estructurado en módulos, que se implementan como hilos de ejecución de tiempo real. Los módulos que se han desarrollado para este robot son:

- Módulo de visión: interpretación visual del entorno.
- Módulo de locomoción: generación de trayectorias.
- Módulo de control: definición de comportamientos.
- DCM: comunicación con sensores/actuadores.

Con esta arquitectura se participó en la "Robocup Mediterranean Open 2010" y en la "Standard Platform League de la Robocup 2010".

3.2.2. Resultados

Cada uno de estos módulos se ha implementado con los siguientes tiempos de ejecución, periodos y prioridades en el Nao V3 y Nao V3+, respectivamente:

Módulo	Prioridad	Periodo (ms)	Carga del procesador (%)
DCM	40	20	20
Locomoción	41	20	20
Control	42	80	5
Visión	43	¿?	< 65%

TABLA 2. PLANIFICACIÓN DE LOS MÓDULOS EN EL NAO V3

Módulo	Prioridad	Periodo (ms)	Carga del procesador (%)
DCM	40	10	40
Locomoción	41	20	20
Control	42	80	5
Visión	43	¿?	< 35%

TABLA 3. PLANIFICACIÓN DE LOS MÓDULOS EN EL NAO V3+

3.2.3. Limitaciones

Al no ser un sistema dedicado, la carga del procesador disponible para el módulo de visión se limita a 65% y 35% en cada caso, lo que definirá su periodo en función del tiempo de procesamiento. Si, además, se incluyera cualquier módulo adicional (por ejemplo, un módulo de autolocalización del robot), la disponibilidad del procesador todavía disminuiría más.

En el caso de la arquitectura distribuida planteada, la carga del DSP disponible para el sistema de visión es siempre del 100% y, además, se libera al procesador principal de uno de los hilos más exigentes. Para no afectar la interacción entre el módulo de visión y el módulo de control se utiliza un medio de comunicación de alta velocidad como es Ethernet.

3.3. ARQUITECTURA PROPUESTA

3.3.1. Plataforma de trabajo: MicroBiro-II

MicroBiro-II es un robot humanoide desarrollado por el Instituto de Automática e Informática Industrial (AI2) de la Universidad Politécnica de Valencia.

FIGURA 24. EL ROBOT MICROBIRO-II

Sus características son las siguientes:

CONSTRUCCIÓN	54 cm y 4 kg 21 grados de libertad
ACTUADORES	Servomotores de distinto par: 16, 28 y 64 Kg/cm Controlados a través del bus RS485 Protocolo del fabricante Dynamixel
UNIDAD DE CONTROL PRINCIPAL	Procesador XScale PXA320 (Monahans) CPU a 800 MHz 1 Gb FLASH, 128 Mb RAM
SOPORTE DE EJECUCIÓN	SO Linux
PERIFÉRICOS	2 conectores USB 2 conectores Ethernet 1 RS232 1 RS485

	1 tarjeta SD CAN Audio
CONJUNTOS DE SENSORES	Sensor de inclinación (módulo acelerómetros) Sensor de presión (módulo galgas) Sensor de visión

Este robot está diseñado para cumplir los requisitos de la “KidSize Humanoid League” de la “Robocup Soccer League”.

Esto conlleva restricciones en el diseño mecánico, en los sensores a utilizar (sólo sensores propioceptivos, excepto el sistema de visión) y en el sistema de visión.

3.3.2. Hardware

(A) Unidad de control principal

El procesador de la unidad de control principal es un Toradex Colibri XScale PXA320 a 806 MHz de muy bajo consumo, basado en el Marvel ARM XScale PXA320 (Monahans).

El procesador PXA320 se integra en la placa modular Toradex Protea para formar el empotrado, con todos los periféricos nombrados anteriormente (2 conectores USB, 2 conectores Ethernet, 1 RS232, 1 RS485, 1 tarjeta SD, CAN Audio. El sistema operativo utilizado es un Linux genérico.

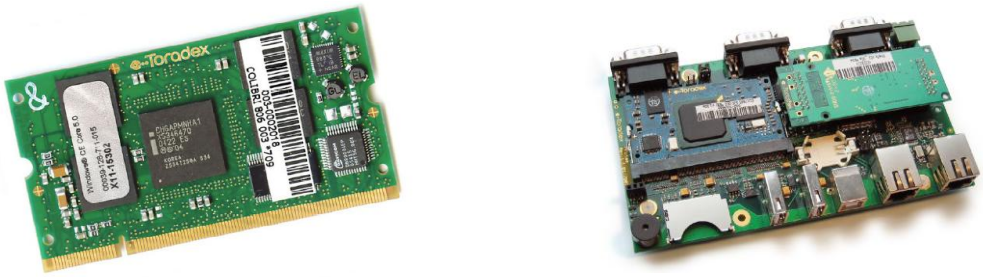


FIGURA 25. TORADEX COLIBRI XSCALE PXA320 I TORADEX PROTEA

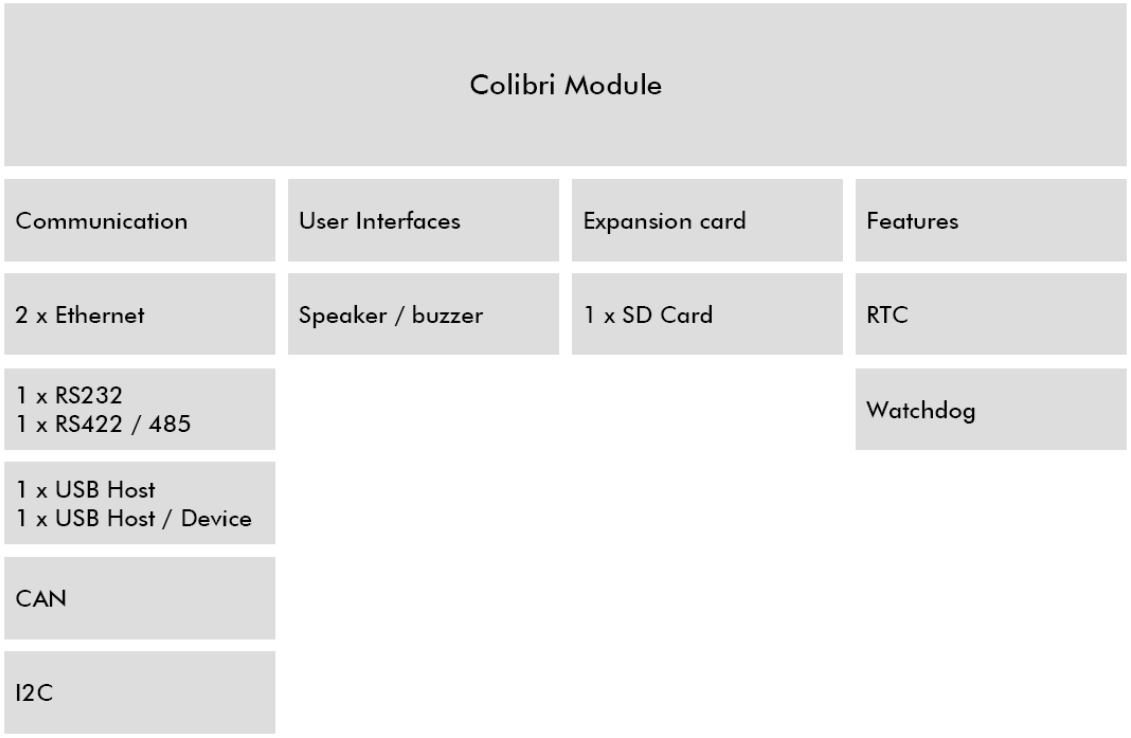


FIGURA 26. PERIFÉRICOS UNIDAD DE CONTROL PRINCIPAL

(B) Sistema de visión

El hardware del sistema de visión está formado por un empotrado basado en un módulo CM-BF537E con los siguientes periféricos:

- Conector Ethernet RJ45 para la comunicación con la unidad de control principal
- Conector serie TTL para la comunicación con los servomotores del cuello.
- Conector para la cámara CMOS.

Como sensor de visión se utiliza una cámara CMOS OV7660 de Omnivision, controlada a través de la interfaz SCCB (Serial Camera Control Bus) y como servomotores los Dynamixel AX-12.

FIGURA 27. HARDWARE DEL SISTEMA DE VISIÓN

Las características del módulo CM-BF537E, que incluye el DSP Blackfin 537 de Analog Devices, se presentan en la Tabla 4.

CPU	Reloj de 600 MHz Arquitectura RISC
MEMORIA	132 Kb de memoria interna (nivel 1) SRAM SRAM/cache de instrucciones y SRAM de instrucciones SRAM/cache de datos más SRAM de datos dedicada Scratchpad SRAM 32 Mb de memoria externa (nivel 3) SDRAM Opciones de arranque flexibles desde distintas zonas de memoria 4 Mb de Flash
PERIFÉRICOS	Interfaz Ethernet Interfaz CAN Interfaz PPI Interfaz SPI 2 puertos serie síncronos (SPORTs) full-duplex 2 UARTs 12 DMAs periféricos 2 DMAs memoria-memoria Manejador de eventos con 32 interrupciones 8 timers/contadores de 32 bits con soporte PWM Core Timer de 32 bits 48 I/O de propósito general (GPIOs), 8 con drivers de alta corriente

TABLA 4. CARACTERÍSTICAS DEL MÓDULO CM-BF537E

3.3.3. Arquitectura de control

En el presente trabajo se propone la integración del sistema de visión en una arquitectura de control distribuida, de forma que todo lo referente a la percepción visual del entorno se realiza a través del sistema de visión activo y la información extraída se comparte con el resto del sistema distribuido a través de la unidad de control principal.

Tal y como muestra la Figura 28, dicha unidad de control es la encargada de interpretar la información de alto nivel proveniente del módulo de locomoción y del sistema de visión y decidir la acción a realizar por ambos módulos.

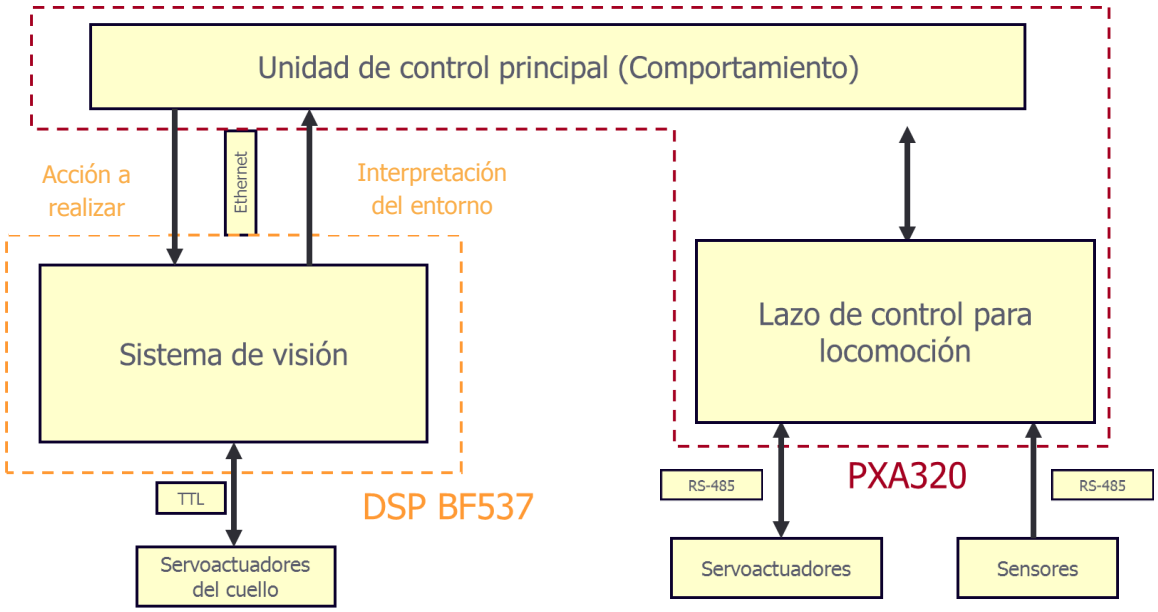


FIGURA 28. ARQUITECTURA DE CONTROL DISTRIBUIDA DEL ROBOT

La arquitectura de control se divide en tres niveles: misión, táctica y reactivo.

- El nivel misión está definido por comportamientos, que están formados por varias habilidades básicas, como por ejemplo buscar la pelota e ir a por ella.
- El nivel táctica está definido por habilidades básicas, que están formadas por trayectorias, como por ejemplo caminar.
- El nivel reactivo define trayectorias o posición de las articulaciones.

El nivel reactivo está incluido en el módulo de locomoción, que procesa los datos recogidos a través de los sensores propioceptivos del robot (módulo de inclinación y módulo de presión) y genera las trayectorias, en función del comportamiento dictado por la unidad de control principal.

El sistema de visión recibe de la unidad de control:

<table>
<tr><td rowspan="2">Acción

a realizar</td><td>Hacer de puente para que la unidad de control principal pueda actuar sobre la posición de los servomotores del cuello.

• Posición del servomotor horizontal.
• Posición del servomotor vertical.</td></tr>
<tr><td>Realizar un comportamiento predefinido

• Buscar pelota, haciéndole tracking.
• Realizar barrido del entorno.
• Mantener posición fija (idle).</td></tr>
<tr><td>DATOS_CD</td><td>Posición y orientación en el espacio del origen de coordenadas del sistema de referencia del sistema de visión, situado en el eje del servomotor horizontal del cuello, extraído de la cinemática directa del robot.

Estos datos son necesarios para el cálculo de la distancia de los objetos por parte del sistema de visión. En concreto, se necesita:

▪ Z_CDM: Posición del centro de masas del robot en la dirección perpendicular al suelo.
▪ Z_HEAD: Posición del origen de coordenadas del sistema de visión respecto del centro de masas del robot en la dirección perpendicular al suelo
▪ OY_HEAD: Rotación del origen de coordenadas en la dirección de movimiento lateral del robot, respecto del centro de masas del robot.
▪ OZ_HEAD: Rotación del origen de coordenadas en la dirección perpendicular al suelo, respecto del centro de masas del robot.</td></tr>
</table>

Cada comportamiento tiene asociado una trayectoria de los servomotores del cuello. Estos son controlados desde el DSP, constituyendo así un sistema de visión empotrado activo.

En el caso del *tracking*, las coordenadas de los objetos en la imagen se traducen a rotaciones de los servos, los cuales se controlan mediante un regulador PID, para poder seguir el objeto en cuestión. Es decir, para realizar el *tracking* de la pelota se utilizan las técnicas del *image-based visual servoing*.

Por otra parte, el sistema de visión envía periódicamente los objetos detectados en la escena y su localización respecto del robot.

El juego de mensajes del protocolo implementado para esto se organiza en estructuras y es:

struct COLIBRI_VISION							
ACCIÓN	**DATOS_CD**						
0x00: Acceso a servomotores 0x01: Comportamientos	SERVO_H_POS [0,1024]	SERVO_V_POS [0,1024]	COMPORTAMIENTO 0x01: TRACK_BALL 0x02: BARRIDO 0x03: IDLE	Z_CDM [0,1024]	Z_HEAD [0,1024]	OY_HEAD [0,1024]	OZ_HEAD [0,1024]
1 byte	2 bytes	2 bytes	1 byte	2 bytes	2 bytes	2 bytes	2 bytes

struct DSP_VISION								
		struct OBJETOS [NO_OBJETOS]						
SERVO_H_POS [0,1024]	SERVO_V_POS [0,1024]	update1	ρ1	φ1	…	upn	ρn	φn
2 bytes	2 bytes	1 byte	2 bytes	2 bytes		1 byte	2 bytes	2 bytes

Cada uno de los n objetos predefinidos tiene un identificador dentro de la estructura OBJETOS. Con el campo “update” se indica si han sido identificados en la última imagen procesada, mientras que su posición se caracteriza con una distancia (ρ) y orientación (Θ) respecto del robot.

3.4. SISTEMA DE VISIÓN

3.4.1. Implementación

La aplicación implementada en el DSP del sistema de visión está concebida como un proceso multihilo: un hilo principal, uno para adquisición de la imagen y uno para interpretación de la imagen.

Los hilos se implementan sobre el VisualDSP Kernel, un sistema operativo muy ligero y robusto, integrado en el entorno de desarrollo del Blackfin que permite, entre otras cosas, definir hilos, prioridades, regiones críticas, semáforos y mensajes.

El hilo principal se encarga de la inicialización de la cámara y las comunicaciones y de lanzar los hilos de adquisición y de interpretación, que se sincronizan entre ellos mediante un mecanismo de semáforos.

El hilo de adquisición de la imagen se encarga de recoger las imágenes que entrega el sensor a través del driver. El hilo de interpretación recoge la imagen guardada en el *buffer* por el hilo de adquisición y aplica los algoritmos para extraer la información requerida. Dicha información es almacenada y comunicada al módulo de control del robot mediante el protocolo sobre UDP creado.

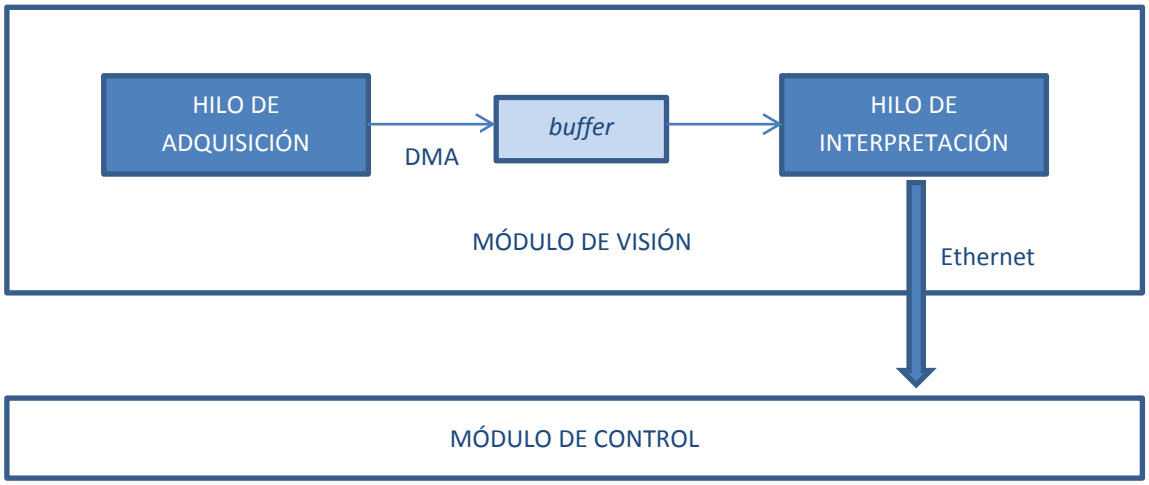


FIGURA 29. IMPLEMENTACIÓN DEL SISTEMA DE VISIÓN

El periodo del hilo de adquisición está ligado al *frame rate* de captura de imágenes de la cámara. Si trabajamos en resolución QVGA (320 x 240 pixels) el *frame rate* es de 30 fps, lo que implica un periodo de 33 ms para el hilo de adquisición. El tiempo de cómputo del hilo de adquisición es mínimo, ya que la imagen se guarda en el *buffer* por DMA (*Direct Memory Access*) sin usar el procesador. Durante este momento, el procesador es ocupado por el hilo de interpretación que, actualmente, tiene un tiempo de cómputo de unos 10 ms.

En conclusión, se obtiene una imagen procesada cada 33 ms, con una utilización de procesador de aproximadamente 30%. El cronograma de tiempos se muestra en la Figura 30 (el hilo superior en la imagen es el hilo de interpretación y el segundo es el hilo de adquisición). La baja carga del procesador permitirá incorporar nuevas fases en el hilo de interpretación sin modificar, de momento, el periodo de adquisición.

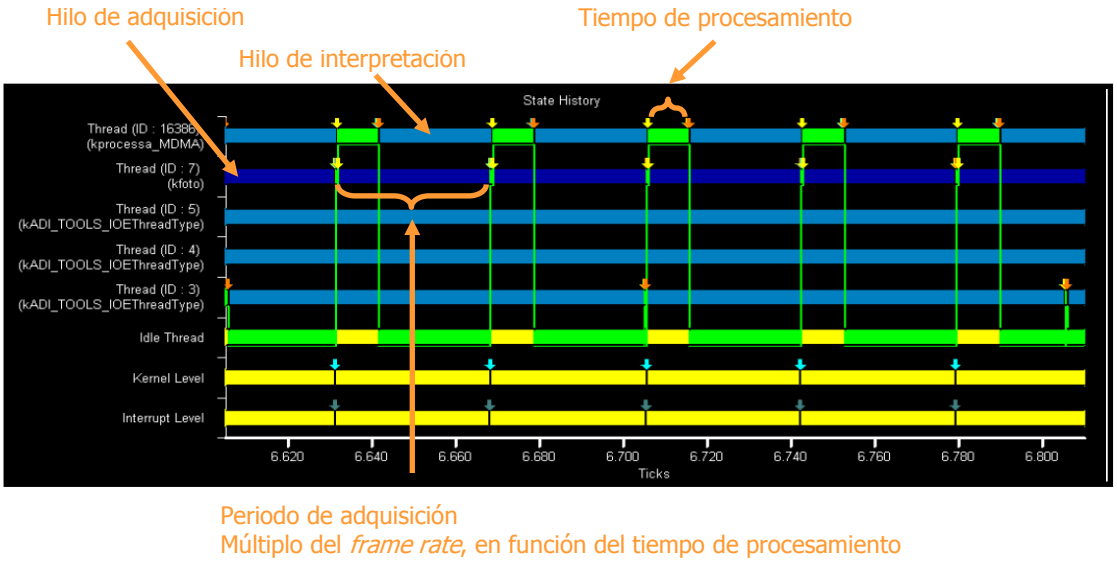


FIGURA 30. CRÓNOGRAMA EJECUCIÓN SISTEMA VISIÓN

3.4.2. Adquisición de la imagen

La adquisición de la imagen se realiza por DMA a través de la interfaz PPI del microcontrolador y se guarda en el *buffer* (objeto compartido con el hilo de interpretación).

La cámara puede proporcionar la imagen en distintos formatos:

- Espacio de color: RGB, grayscale, YUV.
- Resolución: VGA (640 x 480), QVGA (320 x 240), QQVGA (160 x 120).

(A) Espacios de color

Un modelo de color es un modelo matemático abstracto que describe la forma en la que los colores pueden representarse como una tupla de valores, normalmente tres o cuatro, denominados *componentes de color*. Este modelo, junto con una función de mapeado (*gamut*), define un espacio de color. En la Figura 31 se muestran diferentes ejemplos de espacios de color (ProPhoto RGB, sRGB, etc.), basados en el modelo de color RGB.

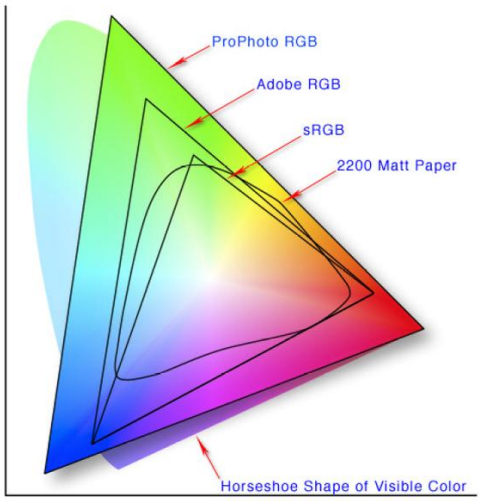


FIGURA 31. ESPACIOS DE COLOR SOBRE EL MODELO RGB

A continuación se presentan los espacios de color utilizados durante el trabajo.

RGB (4:4:4)

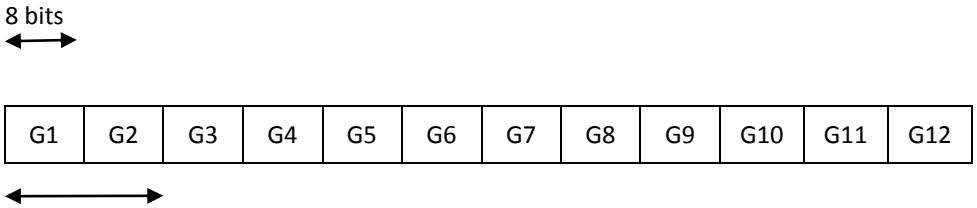
8 bits

R0	G0	B0	R1	G1	B1	R2	G2	B2	R3	G3	B3

2 image pixels

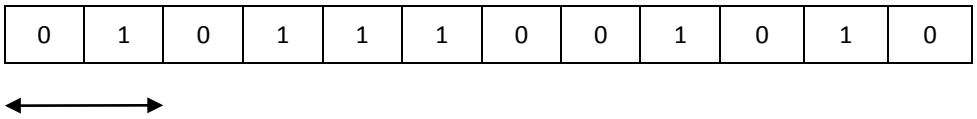
Grayscale or intensity images (0 to 255)

8 bits



BW images (0 or 1)

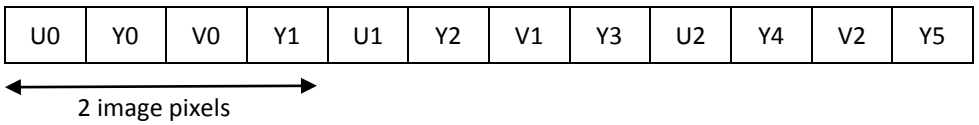
1 bit



2 image pixels

UYVY (4:2:2) colour space

8 bits



El espacio RGB representa cada color como combinación de sus tres componentes: rojo, verde y azul. El espacio YUV, en cambio, separa la información de cada píxel en componentes de iluminancia (Y) y crominancia (U,V).

Teniendo en cuenta que el tiempo de procesamiento es un aspecto crítico, en principio sería preferible tratar la información de color con dos componentes (U,V) en lugar de tres (RGB).

Los rangos de los diferentes componentes del espacio YUV son:

Y = [0,255]
U = [0,255]
V = [0,255]

La Figura 32 muestra una distribución orientativa de los colores en el espacio YUV.

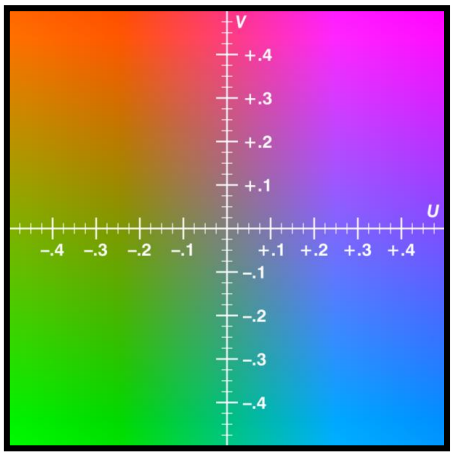


FIGURA 32. VALORES DE UV PARA UN VALOR DE Y = 128

(B) Resoluciones

La cámara puede entregar imágenes en distintas resoluciones, pero no con la misma frecuencia (*frame rate*). Trabajando en VGA se obtienen 15 imágenes por segundo (fps –*frames per second*), mientras que en QVGA y QQVGA se obtienen 30 imágenes por segundo. Es por esto que se ha decidido utilizar la resolución QVGA.

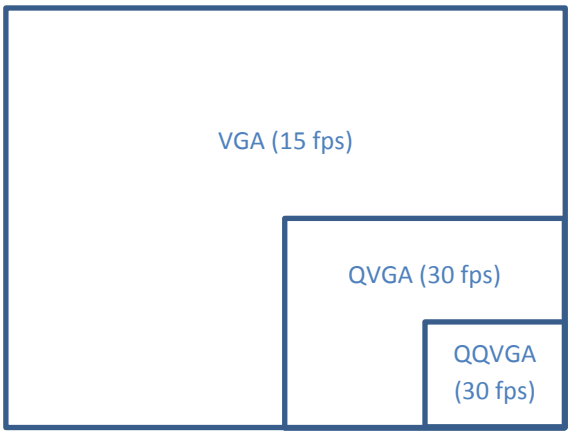


FIGURA 33. ESQUEMA DE RESOLUCIONES DISPONIBLES Y SU *FRAME RATE*

(C) Configuración de los registros de la cámara

Entre las funciones más influyentes de configuración de la cámara para la adquisición de la imagen cabe destacar:

- **Ganancia:** Factor aplicado a cada pixel de la imagen que influye en la intensidad media de la imagen.
- **Exposición:** Tiempo durante el cual la luz está incidiendo sobre el sensor de imagen. Además de influir en la intensidad media de la imagen, es un factor crítico para tomar imágenes en movimiento.

- **Balance de blancos:** Permite conseguir una reproducción de color correcta sin mostrar dominantes de color (rojo, verde o azul), con independencia del tipo de luz que ilumina la escena.

3.4.3. Interpretación del entorno

El trabajo realizado hasta ahora abarca las fases del procesamiento de la imagen y el cálculo de la posición de los objetos respecto del robot.

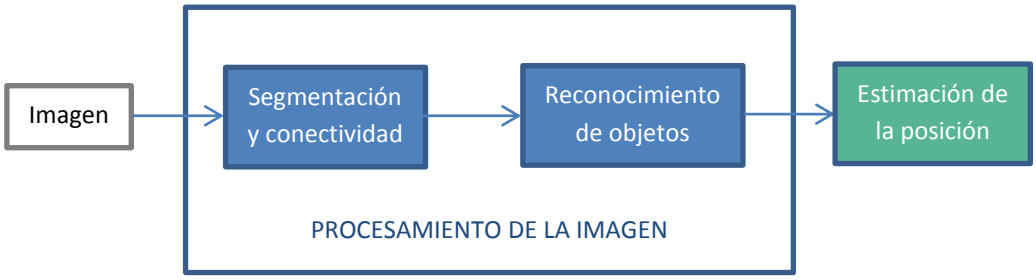


FIGURA 34. PASOS EN LA INTERPRETACIÓN DEL ENTORNO

(A) Procesamiento de la imagen

Los pasos básicos del procesamiento de la imagen se muestran en la Figura 34:

- Segmentación y conectividad de píxeles.
- Reconocimiento de objetos
  - Pelota
  - Balizas

(I) Segmentación y conectividad

El objetivo de las técnicas de segmentación de imagen es extraer los píxeles del conjunto siguiendo unos criterios definidos para simplificar el procesamiento y análisis posterior. Las técnicas de conectividad permiten agrupar píxeles adyacentes similares en entidades más grandes.

En nuestro caso, la caracterización del entorno permite que los criterios de segmentación se basen en el color, de forma que los píxeles extraídos de la imagen corresponden a objetos de colores conocidos con antelación. De la imagen original se aíslan los píxeles cuyo color esté dentro de unos umbrales definidos previamente. Al mismo tiempo, se agrupan los píxeles contiguos del mismo color y se caracterizan las agrupaciones de píxeles (color, número total de píxeles, ancho, alto y centroide).

Para explicar los algoritmos desarrollados se va a utilizar la imagen de la Figura 35.

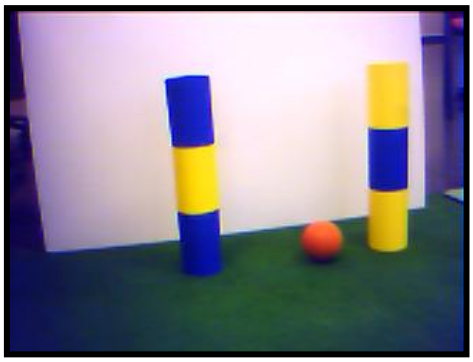


FIGURA 35. IMAGEN A PROCESAR

La calibración de los colores (definición de los umbrales de cada componente del espacio de color) en la imagen se realiza a través de la aplicación gráfica que se explica detalladamente más adelante. Los resultados de la segmentación se muestran en la Figura 36.

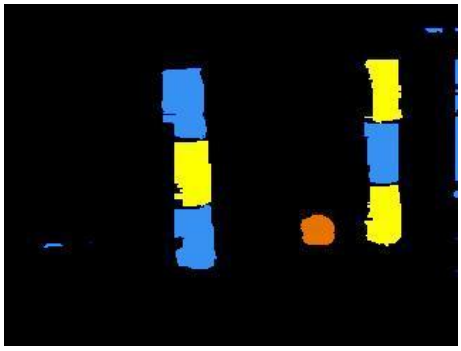


FIGURA 36. SEGMENTACIÓN DE LA IMAGEN

Para la conectividad de píxeles se han comparado distintos algoritmos. Los tiempos de cómputo en Matlab necesarios en cada uno de ellos se muestran en la Tabla 5.

Algoritmo	**t_c (s)**	**% respecto del mejor**
Matrices de adyacencia	1.32	146
Flood fill approach	1.96	216
Equivalence class resolution	0.981	108
Union find	1.04	115
Run length encoding	0.906	100

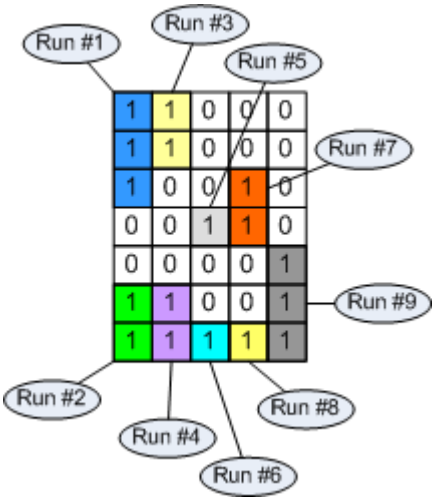
TABLA 5. COMPARATIVA ALGORITMOS DE CONECTIVIDAD

A continuación, se explica detalladamente el algoritmo utilizado, *Run length encoding*.

Considérese como ejemplo básico la siguiente imagen binaria, resultado de la segmentación por color de una imagen

1	1	0	0	0
1	1	0	0	0
1	0	0	1	0
0	0	1	1	0
0	0	0	0	1
1	1	0	0	1
1	1	0	1	1

Puede ser representada como una colección de runs o secuencias de 1s. Trabajando en la dirección de las columnas, dicha imagen contienen las siguientes 9 runs:



Cada *run* puede ser representada por la posición del píxel inicial i por el número de píxeles de la *run*, la *run length*, que dá nombre al algoritmo: *run-length encoding*.

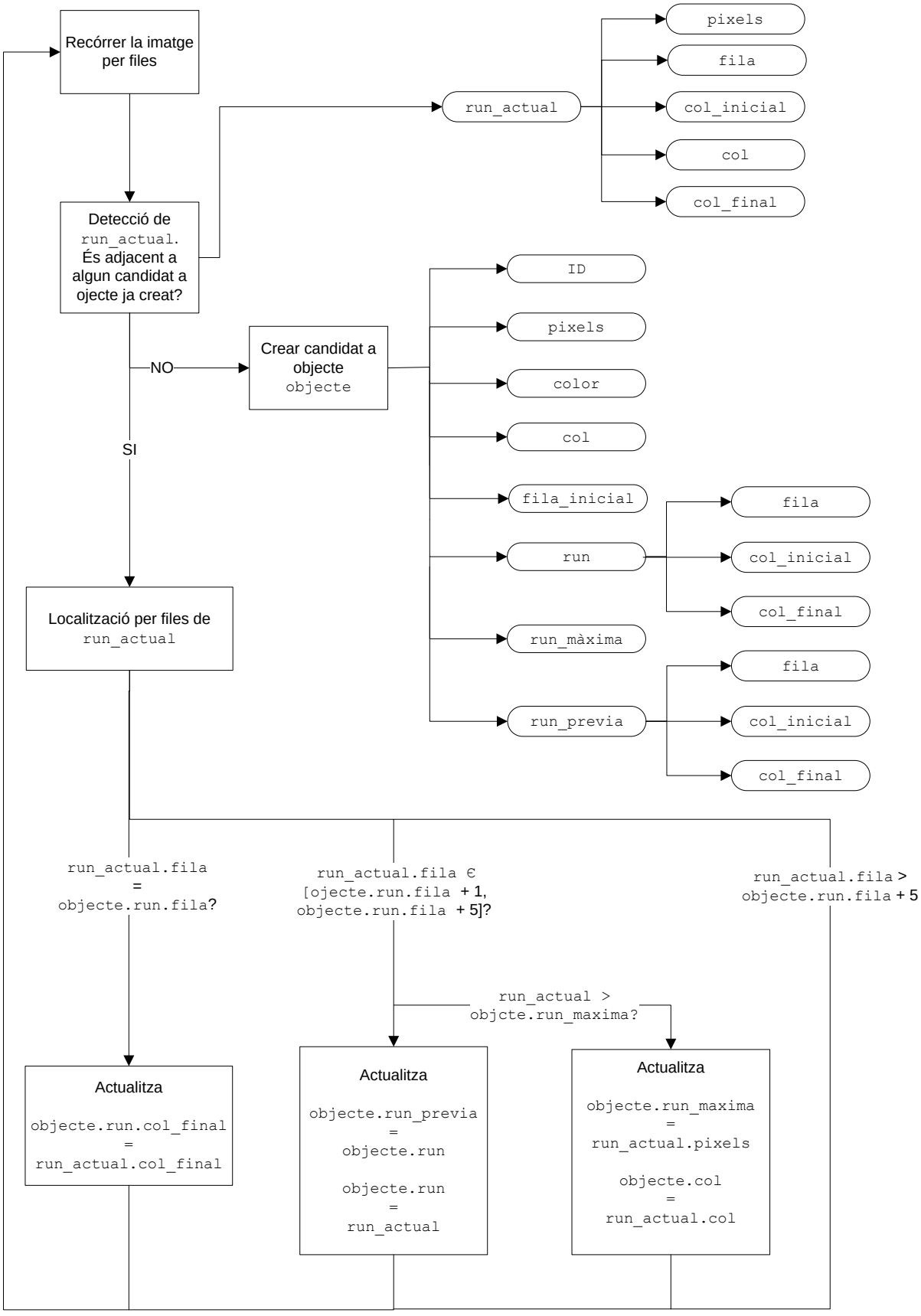
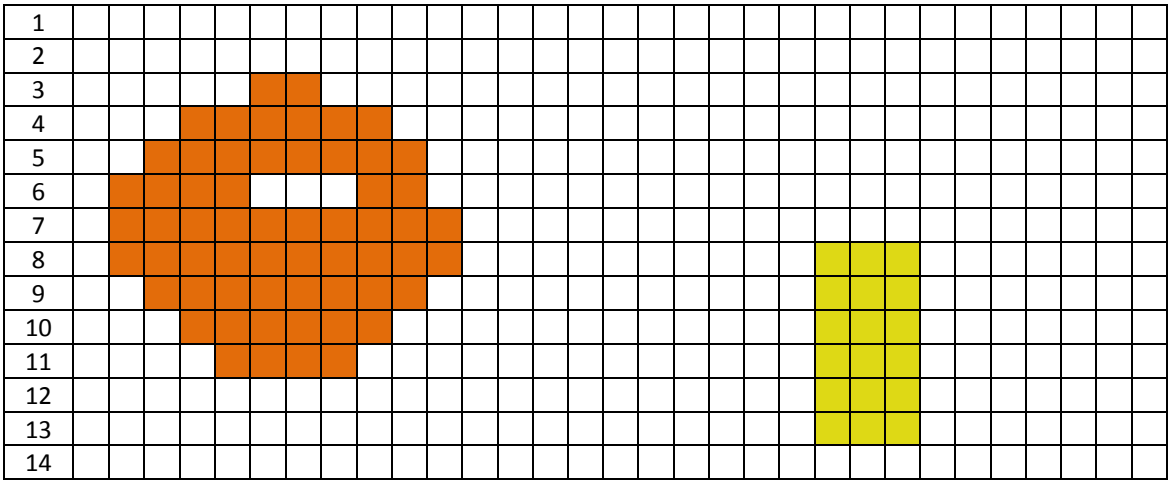


FIGURA 37. ALGORITMO *RUN-LENGTH ENCODING* MODIFICADO

La modificación del algoritmo se centra en la forma en la que se realiza la conectividad entre *runs*, resolviendo así aspectos importantes, tanto a nivel de tiempo de cómputo como de robustez:

- Agrupación de los distintos objetos realizando sólo una pasada por las filas de la imagen.
- Minimización de los efectos de alteración del color: reflexión de la luz en las superficies, sombras, píxeles de diferente color al percibido, etc.

Considerando la siguiente imagen simplificada:



Aplicación del algoritmo de conectividad:

1. Detección de la primera *run* en la fila 3 → creación de candidato a objeto naranja: `objeto_1`

2. Detección de *run* en la fila 4. Adyacente por columnas[1] con `objeto_1.run_previa` (*run* fila 3) → Actualización de `objeto_1.run_previa` *(=run* fila 3) i de `objecte_1.run_actual` *(=run* fila 4)

3. Detección de *run* en la fila 5. Al ser adyacente por columnas con `objeto_1.run_previa` (*run* fila 3) → Actualización de `objeto_1.run_previa` (=*run* fila 4) i de `objeto_1.run_actual` (=*run* fila 5)

4. Detección de *run* en la fila 6. Al ser adyacente por columnas con `objeto_1.run_previa` (*run* fila 4) → Actualización de `objeto_1.run_previa` (=*run* fila 5) i de `objeto_1.run_actual` (=*run* fila 6)

5. Detección de *run* en la fila 6. Al ser adyacente por columnas con `objeto_1.run_previa` (*run* fila 5) → Actualización de `objeto_1.run_actual` (=*run* fila 6)

6. Detección de *run* en la fila 7. Al ser adyacente por columnas con `objeto_1.run_previa` (*run* fila 5) → Actualización de `objeto_1.run_previa` (=*run* fila 6) i de `objeto_1.run_actual` (=*run* fila 7)

[1] Se entiende que dos *runs* son adyacentes por columnas si los píxeles de ambas se solaparan si estuvieran en la misma fila.

7. Detección de *run* en la fila 8. Al ser adyacente por columnas con `objeto_1.run_previa` (*run* fila 6) → Actualización de `objeto_1.run_previa` (=*run* fila 7) i de `objeto_1.run_actual` (=*run* fila 8)

8. Detección de *run* en la fila 8 → creación de candidato a objeto amarillo: `objeto_2.`

9. Detección de *run* en la fila 9. Al ser adyacente por columnas con `objeto_1.run_previa` (*run* fila 7) → Actualización de `objeto_1.run_previa` (=*run* fila 8) i de `objeto_1.run_actual` (=*run* fila 9)

10. Detección de *run* en la fila 9. Al ser adyacente por columnas con `objeto_2.run_previa` (*run* fila 8) → Actualización de `objeto_2.run_previa` (=*run* fila 8) i de `objeto_2.run_actual` (=*run* fila 9)

11. Detección de *run* en la fila 10. Al ser adyacente por columnas con `objeto_1.run_previa` (*run* fila 8) → Actualización de `objeto_1.run_previa` (=*run* fila 9) i de `objeto_1.run_actual` (=*run* fila 10)

12. Detección de *run* en la fila 10. Al ser adyacente por columnas con `objeto_2.run_previa` (*run* fila 8) → Actualización de `objeto_2.run_previa` (=*run* fila 9) i de `objeto_2.run_actual` (=*run* fila 10)

13. Detección de *run* en la fila 11. Al ser adyacente por columnas con `objeto_1.run_previa` (*run* fila 9) → Actualización de `objeto_1.run_previa` (=*run* fila 10) i de `objeto_1.run_actual` (=*run* fila 11)

14. Detección de *run* en la fila 11. Al ser adyacente por columnas con `objeto_2.run_previa` (*run* fila 9) → Actualización de `objeto_2.run_previa` (=*run* fila 10) i de `objeto_2.run_actual` (=*run* fila 11)

15. Detección de *run* en la fila 12. Al ser adyacente por columnas con `objeto_2.run_previa` (*run* fila 10) → Actualización de `objeto_2.run_previa` (=*run* fila 11) i de `objeto_2.run_actual` (=*run* fila 12)

16. Detección de *run* en la fila 13. Al ser adyacente por columnas con `objeto_2.run_previa` (*run* fila 11) → Actualización de `objeto_2.run_previa` (=*run* fila 12) i de `objeto_2.run_actual` (=*run* fila 13)

Aplicando la conectividad a la imagen segmentada de la Figura 36 obtenemos:

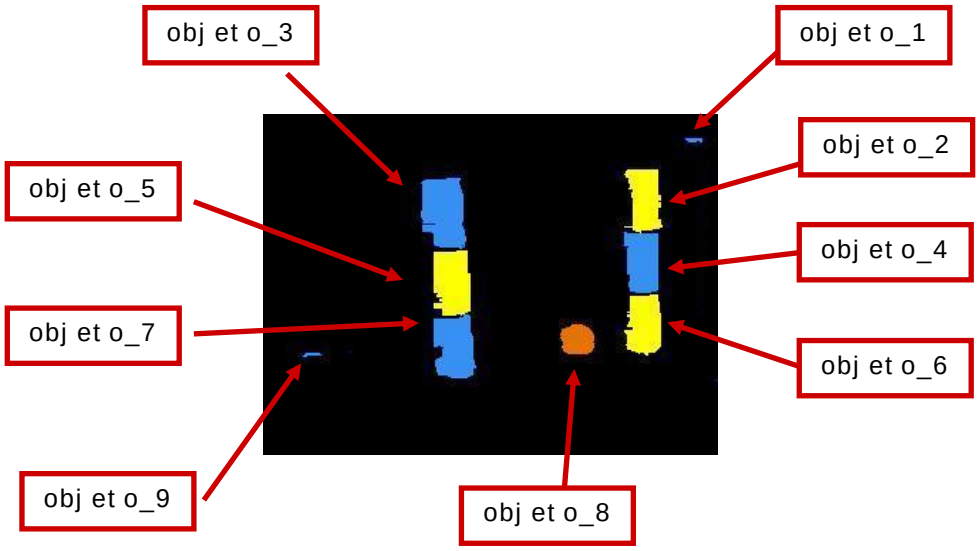


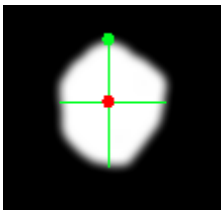
FIGURA 38. CONECTIVIDAD DE PÍXELES

(II) Reconocimiento de objetos

Se aplican filtros de forma para identificar los objetos de interés en la imagen.

PELOTA

Se considera pelota al candidato naranja con mayor número de píxeles que cumpla que la altura y la anchura son aproximadamente iguales (`objeto_8`). Para encontrar el centro de la pelota se utilizan los puntos extremos en la dirección de los ejes.

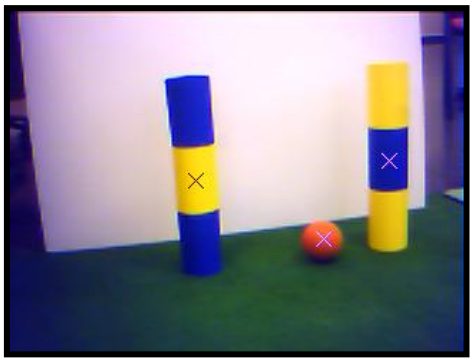


BALIZAS

El reconocimiento de las balizas se basa en la división de estas en tres partes, correspondientes a las tres zonas de color (dos azules y una amarilla o dos amarillas y una azul)

Se considera que representan una baliza aquellos objetos, de los colores citados anteriormente, que tengan el centre aproximadamente en la misma columna de la imagen

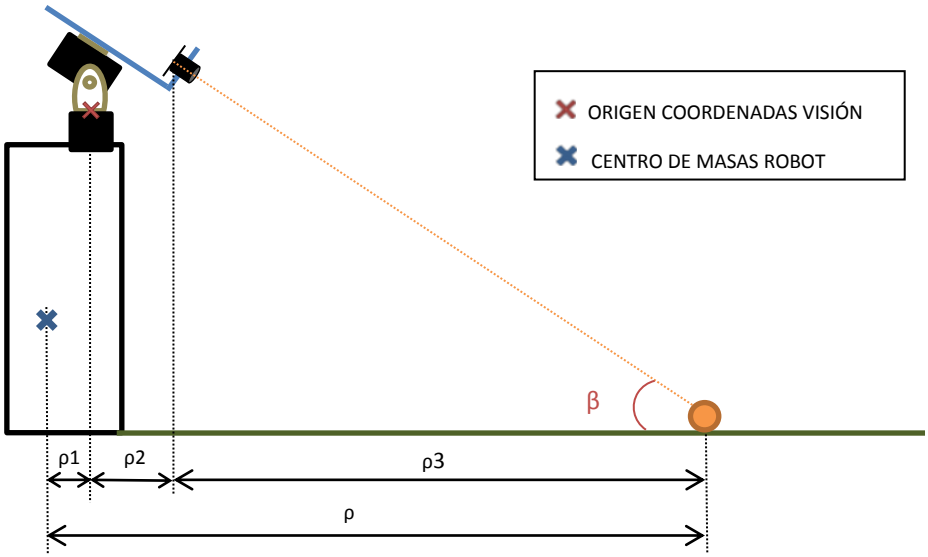
Baliza azul: `objeto_3` + `objeto_5` + `objeto_7`
Baliza amarilla: `objeto_2` + `objeto_4` + `objeto_6`



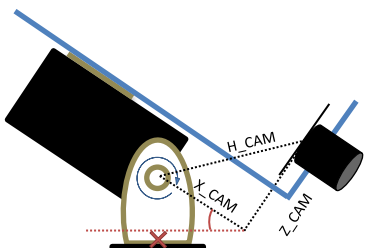
(B) Estimación de la posición

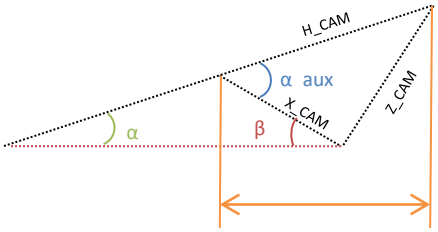
CÁLCULO DE LA DISTANCIA (ρ)

En estos momentos, para tener una buena estimación de la distancia, el objeto debe estar en el centro de la imagen, es decir, hay que hacerle *tracking*. El siguiente paso a realizar para conseguir una mejor estimación, tanto en el centro de la imagen como en cualquier lugar de esta, debería ser la calibración de los parámetros intrínsecos y extrínsecos de la cámara y la distorsión de la lente.



ρ1 tiene en cuenta la inclinación del torso (es decir, la posición del origen de coordenadas del sistema de visión respecto del centro de masas del robot), ρ2 tiene en cuenta la diferencia entre el origen de coordenadas y la situación de la cámara, fruto de la construcción mecánica, y ρ3 la distancia del plano de la imagen al objeto.





$$H\_CAM = \sqrt{X\_CAM^2 + Z\_CAM^2}$$

$$\alpha\_aux = arctg \frac{Z\_CAM}{X\_CAM}$$

$$\boxed{\rho_2 = H_{CAM} \cdot \cos \alpha}$$

$$\boxed{\rho_1 = Z_{HEAD} \cdot \arcsin OY_{HEAD}}$$

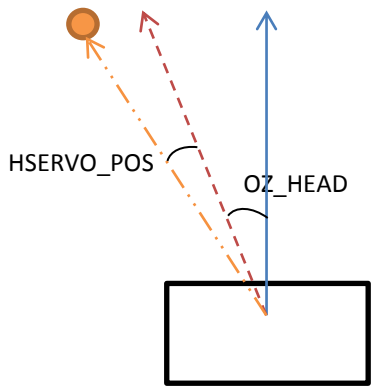
$$\rho_3 = \frac{(Z_{HEAD} + Z_{CDM} + H_{CAM} \cdot \sin(\alpha))}{tg(\beta)}$$

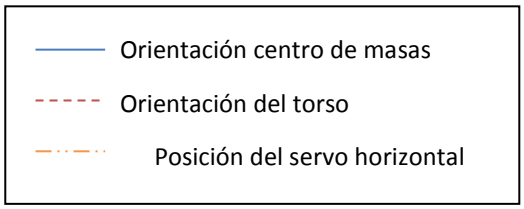
$H_{CAM} \cdot \sin(\alpha)$ es despreciable

$$\boxed{\rho_3 = \frac{(Z_{HEAD} + Z_{CDM})}{tg(\beta)}}$$

CÁLCULO DE LA ORIENTACIÓN (Θ)

La orientación está relacionada con el giro del torso OZ_HEAD y con la posición del servo horizontal.





(C) Comunicación con la unidad de control principal

Se transmite a la unidad de control principal la posición de los objetos identificados respecto del robot a través de Ethernet UDP, utilizando el protocolo especificado en el apartado Arquitectura de control.

3.5. APLICACIÓN GRÁFICA (GUI)

La experiencia acumulada durante el desarrollo del trabajo y, sobretodo, durante la preparación y participación en distintos eventos Robocup (entre ellos la *Standard Platform League* de la *Singapore Robocup 2010*), ha llevado a la conclusión que es absolutamente necesario tener una herramienta software que permita adaptarse rápidamente a las condiciones de iluminación del entorno.

Con este fin, se ha desarrollado una aplicación gráfica que, a través de una conexión Ethernet, permite configurar los parámetros de la cámara y calibrar los colores de los objetos de interés de forma rápida.

La interfaz gráfica está organizada en tres grandes bloques:

- Conexión: Establece una conexión TCP/IP con el sistema de visión del robot.

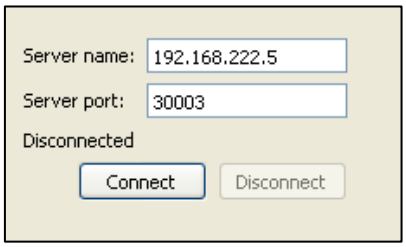


FIGURA 39. CUADRO DE CONEXIÓN DE LA GUI

- Calibración: Define los umbrales (*thresholds*) de color de cada objeto a identificar (*ball, yellow goal, blue goal, lines, carpet*), a partir de la imagen capturada.

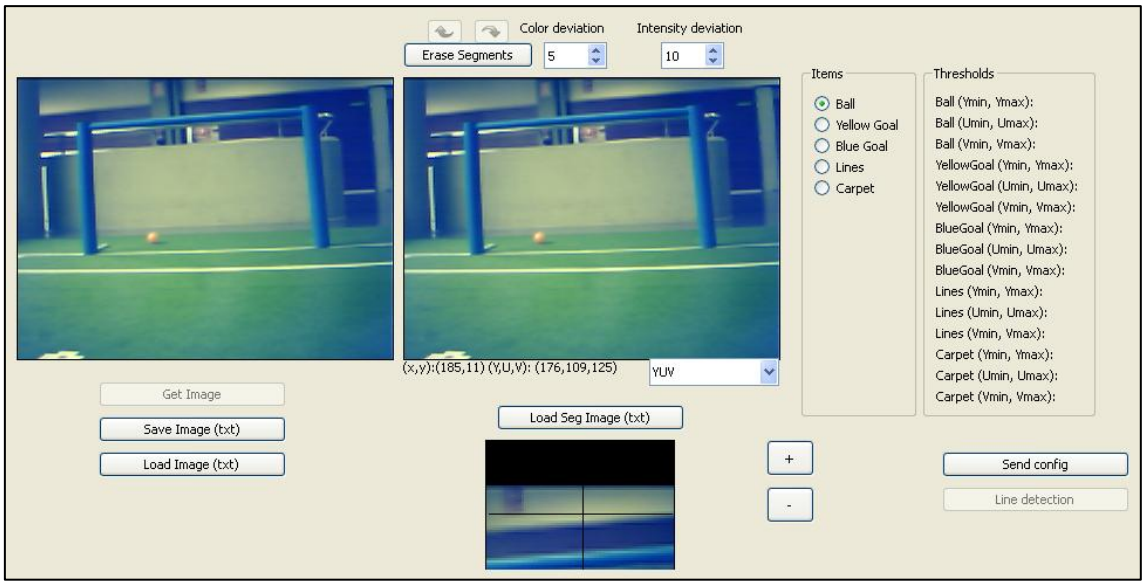


FIGURA 40. CUADRO DE CALIBRACIÓN GUI

Un ejemplo de calibración del color de la portería azul se muestra en la Figura 41.

Los píxeles de la portería se van marcando con el ratón. Cada vez que se marca un píxel, se resaltan aquellos que están dentro de los rangos, *color deviation* e *intensity deviation*. Al final del proceso, en las etiquetas *Thresholds* se anotan los límites mínimos y máximos de cada uno de los componentes del espacio de color. En este caso (Ymin, Ymax) = (53,164); (Umin, Umax) = (140,186); (Vmin, Vmax) = (75,95).

Estos valores se utilizarán en la fase de segmentación de la imagen durante el procesamiento en el DSP. La imagen segmentada estará compuesta por los píxeles resaltados en la GUI.

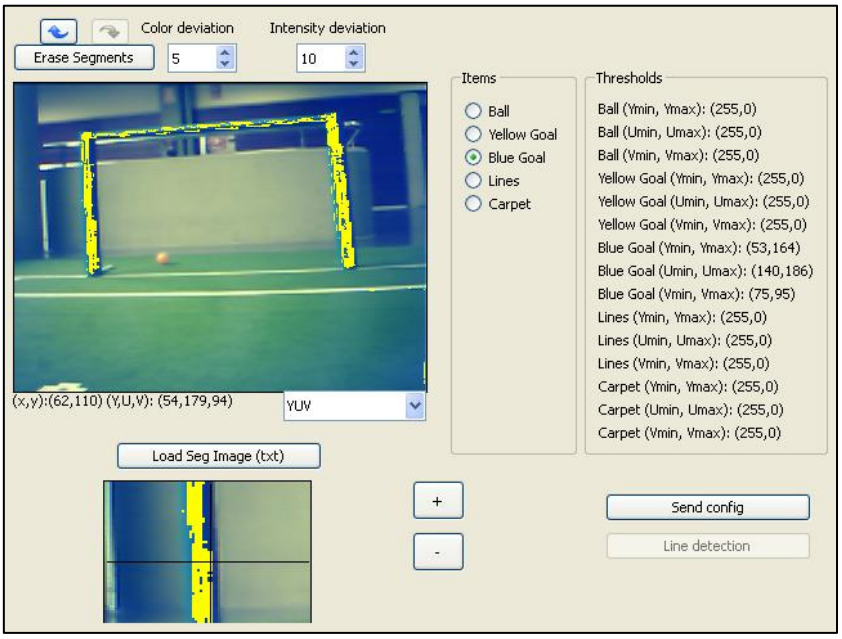


FIGURA 41. EJEMPLO DE CALIBRACIÓN DE LA PORTERÍA AZUL CON LA GUI

- Configuración de los parámetros de la cámara: Adaptación a las condiciones de iluminación. Se varía, principalmente, el tiempo de exposición para la toma de la imagen y las ganancias de los píxeles y de los canales RGB.

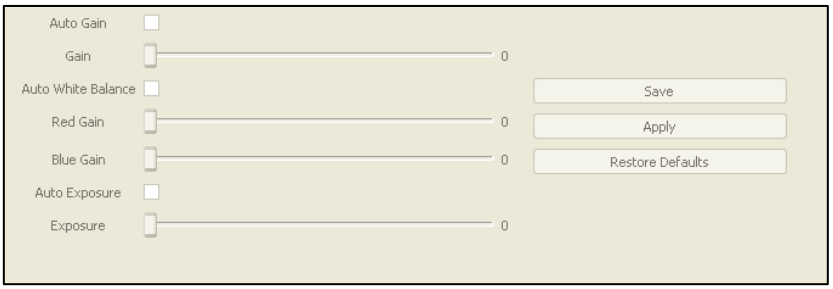


FIGURA 42. CUADRO DE CONFIGURACIÓN DE LOS PARÁMETROS DE LA CÁMARA

Se ha implementado un juego de mensajes para modificar on-line los parámetros de la cámara y adquirir las imágenes.

ACCIÓN	CMD
Tomar nueva imagen	'i'
	1 byte

ACCIÓN	CMD	CMD2	VALOR
Modificar parámetros de la cámara	'p'	'G' – Auto Gain 'W' – Auto White Balance 'a' – Auto Exposure	ENABLE / DISABLE
	1 byte	1 byte	1 byte

ACCIÓN	CMD	CMD2	VALOR
Modificar parámetros de la cámara	'p'	'g' – ganancia cámara 'r' – ganancia canal rojo 'b' – ganancia canal azul	[0,255]
	1 byte	1 byte	1 byte

ACCIÓN	CMD	CMD2	VALOR
Modificar parámetros de la cámara	'p'	'e' – exposición	[0,500]
	1 byte	1 byte	2 bytes

Los resultados de la calibración se envían a través de la conexión TCP/IP al sistema de visión del robot. Los datos se almacenan en la memoria Flash del DSP y se cargan al iniciar el programa.

- Control de los servomotores de la cabeza.

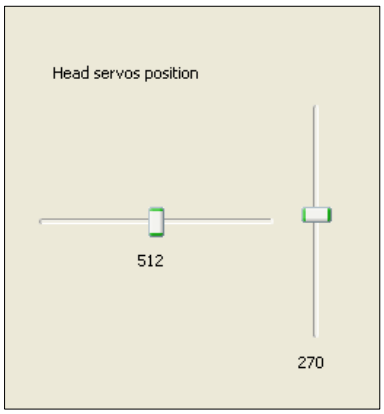


FIGURA 43. CUADRO DE CONTROL DE LOS SERVOMOTORES DE LA CABEZA

ACCIÓN	CMD	CMD2	VALOR
Modificar posición servo horizontal	'p'	'1'	[0,1024]
	1 byte	1 byte	2 bytes

ACCIÓN	CMD	CMD2	VALOR
Modificar posición servo vertical	'p'	'2'	[0,1024]
	1 byte	1 byte	2 bytes

Además de esto, esta aplicación puede servir de plataforma de testeo de algoritmos, pudiéndose trasladar rápidamente al DSP, al usar ambos lenguaje C++.

3.6. EJEMPLO DE APLICACIÓN

En este apartado se presenta a modo de ejemplo un comportamiento del robot: localizar la pelota, ir hasta ella y chutar.

Como se ha presentado en el apartado 3.3.3, la arquitectura de control está dividida en tres niveles: misión, táctica y reactivo. En este caso:

Se le indica al sistema de visión el comportamiento a realizar (TRACK_BALL) y se le manda los datos extraídos de la cinemática directa (DATOS_CD) necesarios para el cálculo de la posición de los objetos al inicio y cada vez que el robot se mueve. El sistema de visión devuelve a cada ciclo de ejecución (40 ms) la localización de los objetos encontrados en las imágenes en la estructura DSP_VISION.

MISIÓN: Comportamiento GOTOBALL. Ir a por la pelota y chutar.

TÁCTICA: Máquina de estados.

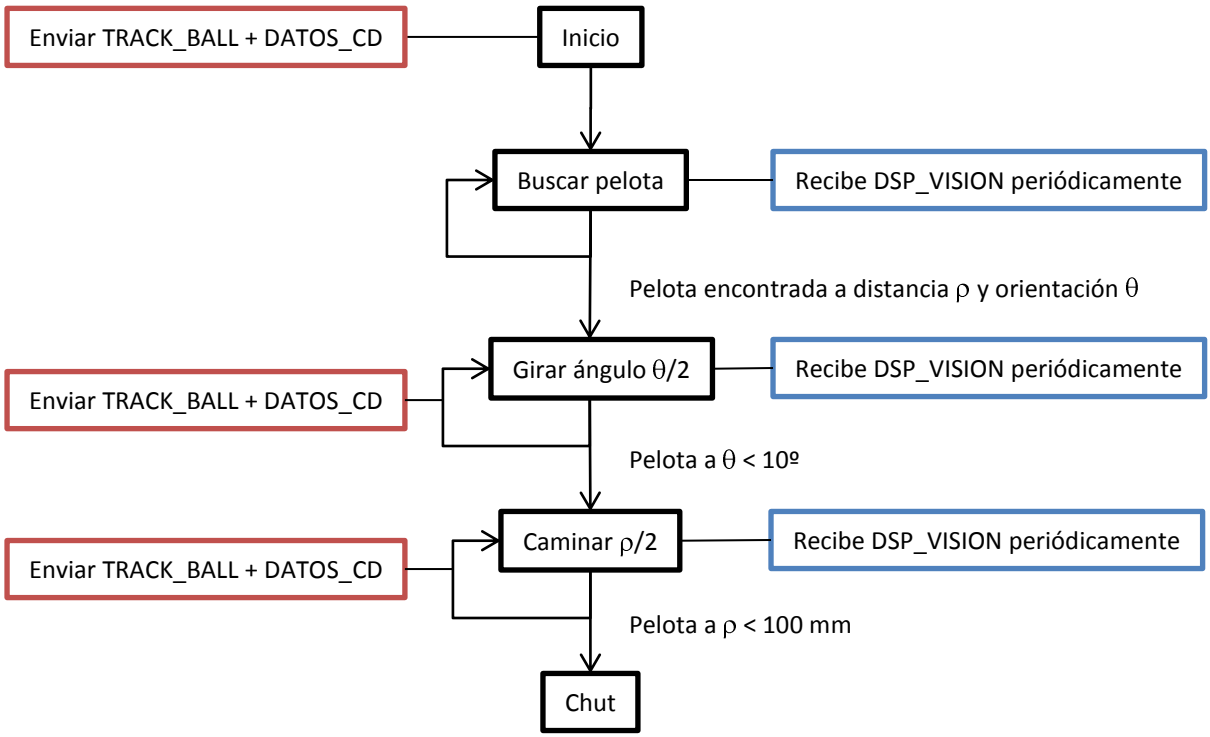


FIGURA 44. MÁQUINA DE ESTADOS DEL COMPORTAMIENTO GOTOBALL

El vídeo de este ejemplo de aplicación se incluye en la documentación aportada.

4. CONCLUSIONES

4.1. CONLCUSIÓN

El trabajo desarrollado ha permitido dotar al robot humanoide MicroBiro-II de un sistema de visión activo empotrado.

De la aportación hecha cabe destacar:

- Se ha definido e implementado una arquitectura distribuida que dota al robot de una capacidad de procesamiento muy elevada.
- Es un sistema de visión activo, lo que implica que es el propio sistema de visión el que decide los movimientos a realizar en función de su interpretación del entorno.
- Se ha dotado al sistema de visión de la capacidad de interpretar una parte de un entorno cambiante tipo Robocup.
- Se ha diseñado de tal forma que la interpretación de nuevos entornos se reduce a la implementación de la algorítmica necesaria dentro de uno de los hilos de la aplicación del DSP.
- Se ha desarrollado una interfaz gráfica de control del sistema de visión, capaz de posicionar los motores, modificar los parámetros de la cámara y calibrar los colores de los objetos

Un artículo sobre el trabajo desarrollado (1) ha sido presentado en el Workshop de Agentes Físicos (WAF) del Congreso Español de Informática (CEDI) 2010. Además, ha sido seleccionado para la revista sobre el workshop (JOPHA – Journal of Physical Agents).

4.2. TRABAJO FUTURO

La realización de este trabajo abre muchas opciones de líneas de investigación futuras.

A nivel del sistema de visión:

- Ampliar los algoritmos de interpretación del entorno Robocup para poder localizar líneas del campo y robots.
- Estudiar la posibilidad de utilizar un sistema de visión estéreo.

A nivel del robot humanoide:

- Autolocalización del robot, fusionando la información proveniente del sistema de visión y de la odometría.
- Desarrollo de comportamientos a partir de la interpretación del entorno hecha por el sistema de visión.

A nivel de coordinación de robots:

- Fusión y compartición de datos de interpretación del entorno y autolocalización entre robots.
- Desarrollo de estrategias conjuntas de los robots.

5. TRABAJOS CITADOS

1. *Sistema de visión empotrado en arquitectura de control distribuida para robot humanoide.* **Muñoz, Pau, Blanes, J.F. y otros.** 2010. Workshop de Agentes Físicos 2010.

2. *Real Time control of 20 DoF humanoid robot.* **Muñoz, M. Blanes, J.F., Albero, M. y otros.** 2009. IFAC Workshop on Real-Time Programming and International Workshop on Real-Time Software.

3. *Arquitectura y desarrollo del robot humanoide microBIRO-II.* **Muñoz, M., Arjona, J., Coronel J.O., Blanes J.F., y otros.** 2009. XXX Jornadas en Automática.

4. *Visual Servoing for Dual Arm Motions on a Humanoid Robot.* **Vahrenkamp, N., Böge, C., Welke, K., Asfour, T., Walter, J., Dillmann, R.** 2009. IEEE/RAS International Conference on Humanoid Robots.

5. *Visually-Guided Grasping while Walking on a Humanoid Robot.* **Mansard, N., Stasse, O., Chaumette, F., Yokoi, K.** 2007. IEEE International Conference on Robotics and Automation.

6. **Sabe, K., Fukuchi, M. y otros.** *Obstacle avoidance and path planning for humanoid robots using stereo vision .*

7. *Stair climbing for humanoid robots using stereo vision.* **Gutmann, J.S., Fukuchi, M., Fujita, M.** 2004. IEEE/RSJ International Conference on Intelligent Robots and Systems.

8. *Vision-guided humanoid footstep planning for dynamic environments.* **Michel, P., Chestnutt, J., Kuffner, J., Kanade, T.** 2005. IEEE-RAS International Conference on Humanoid Robots.

9. *Locomotion planning of humanoid robots to pass through narrow spaces.* **Kanehiro, F., Hirukawa, H.** 2004. Proceedings of the IEEE International Conference on Robotics and Automation.

10. *aiRobots: Team Description for Humanoid KidSize League of RoboCup 2010.* **otros, Tzuu-Hseng S. y.** 2010.

11. *Persia Humanoid Robot. Team Description Paper 2010.* **otros, S. Hamidreza Mohades Kasaei y.** 2010.

12. *Cyberlords RoboCup 2010 Humanoid Size Team Description Paper.* **Lupián, L. F., y otros.** 2010.

13. *Darmstadt Dribblers. Team Description Paper for Humanoid KidSize League of RoboCup 2010.* **Firedmann, M. y otros.** 2010.

14. *Robo-Erectus Jr - 2010 KidSize Team Description Paper.* **otros, Buck Sin N. y.** 2010.

15. *Team DARWIn. Team Description for Humanoid KidSize League of Robocup 2010.* **otros, Jaekweon Han y.** 2010.

16. *Team Description 2010 for Team RO-PE.* **otors, Soo Theng Koay y.** 2010.

17. *ZJUDancer Team Description Paper.* **otros, Tang Qing y.** 2010.

18. *Team Description Paper: HuroEvolution Humanoid Robot for RoboCup 2010 Humanoid League.* **otros, Chung-Hsien Kuo y.** 2010.

19. *Chibi Dragon Team Description Paper.* **otros, Aphilux Buathong y.** 2010.

20. *FUmanoid Team Description Paper 2010.* **otros, Bennet Fischer y.** 2010.

21. *Team NYP Lions: Team Description Paper.* **otros, Lim Sock Lip y.** 2010.

22. *PKU-SHRC Team Description for RoboCup 2010.* **otros, Guangnan Ye y.** 2010.

23. *RoboPatriots: George Mason University 2010 RoboCup Team.* **otros, Keith Sullivan y.** 2010.

24. *WF Wolves KidSize Team Description RoboCup 2010.* **Gerndt, R. y otros.** 2010.

25. *Team BSRU-I: Team Description Paper.* **otros, Chokchai Pengyasa y.** 2010.

26. *PIONEROS MEXICO Team Description Paper RoboCup 2010 Singapore.* **Ramírez Márquez, R.C. y otros.** 2010.

27. *Bogobots-TecMTY humanoid kid-size team 2010.* **Villareal-Pulido, G. y otros.** 2010.

28. *Robust and efficient embedded vision for Nao in Robocup.* **Cavestany, P., Muñoz, P., Herrero-Pérez, D., Blanes, J. F., Martínez-Barberá, H.** Marzo 2010. XI Workshop en agentes físicos. Congreso Español de Informática.