PROYECTO DE SISTEMAS INFORMÁTICOS

CURSO 2009/2010

PLANIFICACION DE TRAYECTORIAS PARA UN ROBOT MOVIL

Autores:

Miguel Ángel Fernández Lancha
David Fernández Sanz
Carlos Valmaseda Plasencia

Director:

José Jaime Ruz Ortíz

AUTORIZACIÓN DE LOS AUTORES

Por la presente se autoriza a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, la documentación y el prototipo desarrollado.

Los autores:

Miguel Ángel Fernández Lancha

David Fernández Sanz

Carlos Valmaseda Plasencia

RESUMEN

Las posibilidades que ofrecen los robots móviles son muy importantes en los campos militar, sanitario, de expediciones terrestres con fines académicos o civiles, en la inspección de instalaciones nucleares, y en general en todo tipo de entornos hostiles y de seguridad.

En el presente proyecto hemos desarrollado un sistema que controla los desplazamientos de un robot móvil SRV-1 que dispone de un módulo de visión estereoscópica. La aplicación se ejecuta en un computador que se comunica con el robot a través de Wi-Fi. Para facilitar el diseño, nuestro primer paso fue desarrollar un simulador 3D. En la memoria explicamos el proceso seguido en el desarrollo del simulador y el diseño de las trayectorias del robot.

Dado el carácter abierto de la especificación del proyecto y la necesidad de coordinación con otro grupo encargado del procesamiento de las imágenes estereoscópicas, optamos por utilizar en el desarrollo del sistema una metodología de tipo OpenUp[1] que facilita la incorporación de prestaciones surgidas en curso del desarrollo y afrontar problemáticas similares en futuros proyectos.

Palabras clave: Robots móviles, planificación de trayectorias, evasión de obstáculos, simulación 3D, navegación.

ABSTRACT

The potential of mobile robots is very important in military camps, health, land expeditions for academic or civilian, in the inspection of nuclear facilities, and generally in all types of hostile environments and security.

In this project we have developed a system that controls the movement of an SRV-1 mobile robot that has a stereo vision module. The application runs on a computer that communicates with the robot via Wi-Fi. To facilitate the design, our first step was to develop a 3D simulator. In the report explained the process followed in developing the simulator and the design of robot trajectories.

Given the open nature of the project specification and the need for coordination with other group responsible for the processing of stereoscopic images, we chose to use a system development methodology OpenUP type that facilitates the incorporation of benefits arising in course of development and face similar problems in future projects.

Keywords: Mobile robots, path planning, obstacle avoidance, 3D simulation, navigation.

INDICE

1.-INTRODUCCIÓN

La robótica móvil es una valiosa herramienta para explorar entornos inaccesibles al ser humano por su lejanía, coste o peligrosidad y para realizar tareas desagradables o laboriosas. Es un campo relativamente nuevo, hasta hace poco experimental, pero que ya se está aplicando a problemas reales con resultados satisfactorios.

Una mayor autonomía del robot, es decir, una menor supervisión humana para su correcto funcionamiento es uno de los propósitos primordiales de esta área de conocimiento. Lograr dicha meta plantea numerosos desafíos. El más obvio es la propia locomoción que ha de complementarse con la percepción del entorno, el posicionamiento en él, el cálculo de trayectorias y la navegación de éstas.

La simulación 3D juega un rol relevante tanto en el diseño como en el control de robots móviles. Permite reflejar la situación del robot real o simular escenarios hipotéticos. A la vez, suministra una interfaz gráfica para el control del robot.

Durante el curso académico 2009/10 se nos ha brindado a los integrantes de este grupo la oportunidad de sumergirnos en esta área de conocimiento interdisciplinar de la mano del Surveyor SRV-1. Se trata de un robot autónomo tipo tanque de dimensiones reducidas, controlado remotamente a través de WiFi. El modelo utilizado incorpora un módulo para la visión estereoscópica.

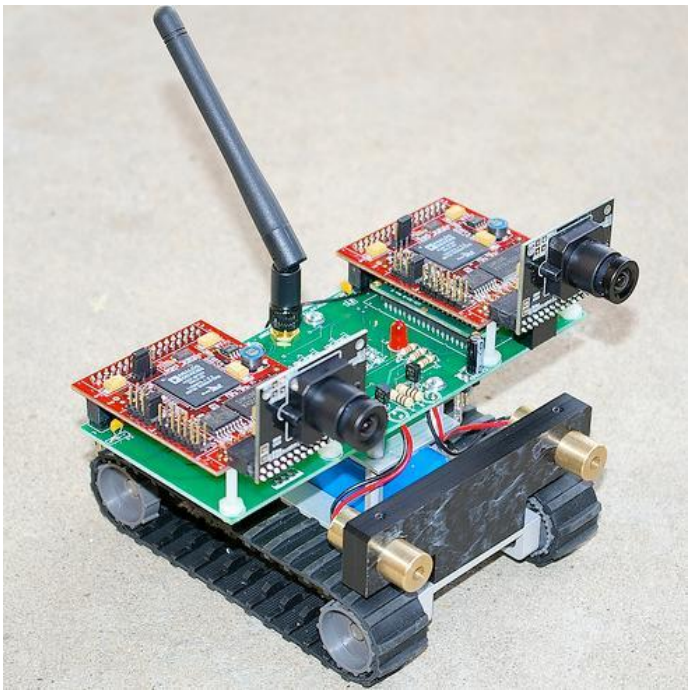


Ilustración 1: Robot móvil Surveyor SRV-1 con módulo para la visión estereoscópica.

En el presente proyecto hemos diseñado e implementado un sistema cuyo propósito principal es planificar trayectorias para este robot móvil.

El sistema se compone de dos partes diferenciadas: una parte hardware y otra software. Los dispositivos físicos que constituyen la parte hardware son el robot móvil, un computador y la conexión Wi-Fi entre los mismos. La parte software incluye la herramienta desarrollada, un sistema operativo sobre el que se ejecuta y una serie de librerías que nos permiten la comunicación con el robot, ya sea por medio de instrucciones de conexión, movimiento o captura de imágenes. La aplicación se ejecuta sobre el computador el cual conecta con el robot móvil a través de Wi-Fi.

En cuanto al robot móvil utilizado es el modelo SRV-1 con sistema de visión estereoscópica de la compañía Surveyor. El sistema de visión estereoscópica se compone de dos cámaras, cada una de ellas incluye su propio procesador.

La aplicación se ejecuta en .NET Framework sobre el sistema operativo Windows. Está implementada en el lenguaje de programación C# y se ha desarrollado en el entorno de programación Microsoft Visual Studio con la ayuda del motor gráfico TrueVision 3D.

La herramienta realiza la planificación de trayectorias entre la coordenada inicial (situación donde está el robot) y una coordenada de destino que le sea indicada. Además simula planificaciones de caminos con escenarios previamente cargados con obstáculos o también tenemos la posibilidad de ir reconociendo ese terreno mediante el análisis de imágenes por medio de la visión estereoscópica.

Por ello, el usuario tendrá la posibilidad de ver en tiempo real en qué situación está el robot y lo que puede ver desde sus cámaras en 1ª persona. La planificación de los caminos se hace en base a la información que tenemos del entorno de la simulación.

En cuanto a la planificación podemos decir que consiste en encontrar un camino que evite peligros de colisión con un algoritmo de búsqueda A* teniendo en cuenta las distancias de seguridad establecidas en el campo de la robótica. Ese camino se transforma en listas de acciones para la simulación y para el robot SRV-1. Cuando queremos transmitir esas acciones al robot se enviaran vía WiFi por medio de las librerías Aforge, de las que hemos hablado anteriormente, la cuales nos permiten la comunicación bidireccional con el robot.

En cuanto al entorno gráfico de la herramienta podemos destacar el simulador tridimensional, el cual nos permite ver con claridad y sencillez el escenario, la situación del robot y evaluar la calidad de la trayectoria generada. Hay que destacar que el simulador se comporta como el robot real, es decir, no tiene ninguna referencia de orientación. Los giros se hacen por tiempos de la misma forma que el

robot SRV-1, lo que nos permite más realismo a la hora de la ejecución de la simulación. Ésto es importante porque dependiendo del ajuste que se dé al robot en cuanto a fuerza de ruedas, peso, fricción…podríamos ajustarlo para cualquier tipo de terreno y tener una precisión simulador-realidad mayor.

La herramienta también está preparada para tener un sistema de referencia del SRV-1 mediante GPS que en este caso se simularía con un sistema de cámaras que nos proporcionaría la posición del robot en coordenadas relativas enviadas a través del protocolo de transmisión UDP. Un sistema como éste nos proporciona mayor precisión puesto que al no tenerlo estamos moviéndonos con una navegación estimada.

A parte de que la herramienta tenga la posibilidad de la planificación de caminos con un entorno conocido o sin conocer, damos la posibilidad de comunicar directamente con el robot y enviarle directamente instrucciones de movimiento, captura de imágenes y video en tiempo real.

Debido a la envergadura del proyecto, éste fue divido en dos partes. La parte correspondiente a nuestro grupo se centra en la estrategia de planificación de rutas mientras que la del otro grupo se encargaría del tratamiento de imágenes. Para no depender de éste se decidió crear un algoritmo de detección de objetos por colores mediante la disparidad.

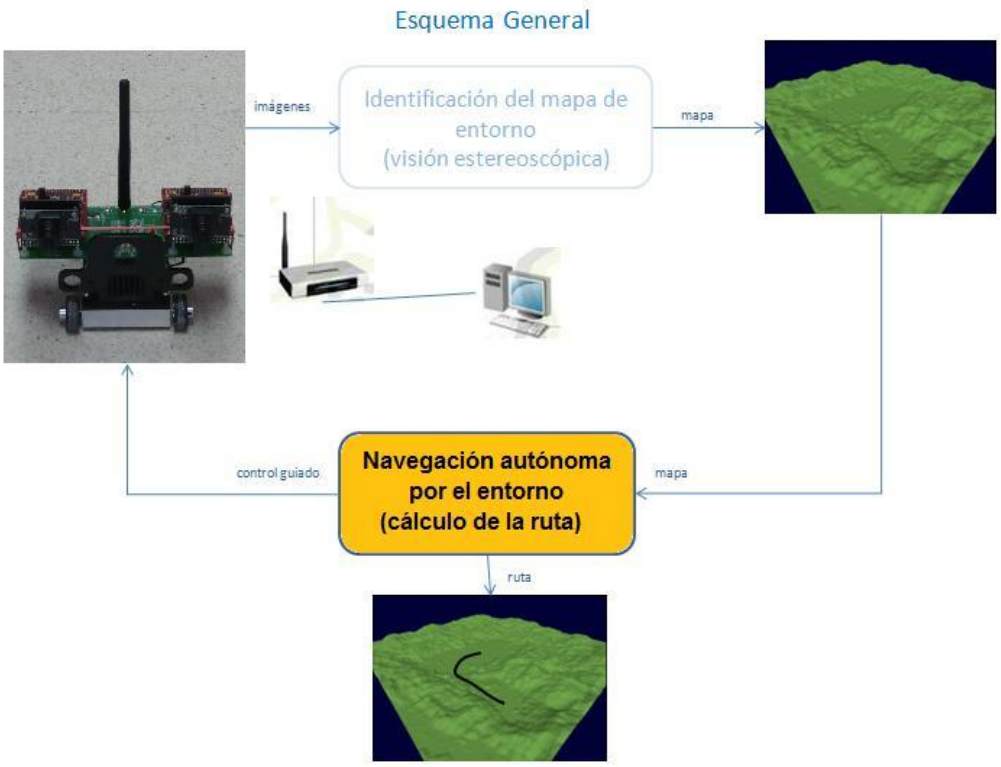


Ilustración 2: División del proyecto. La parte correspondiente a nuestro grupo es la navegación autónoma por el entorno (cálculo de ruta).

Desde un primer momento creímos conveniente enfocar el proyecto desde el paradigma de desarrollo ágil de software tomando como referente una metodología Open Unified Process (OpenUP) debido a la inclusión de objetivos según avanzábamos.

En los siguientes apartados veremos cómo se ha ido desarrollando el proyecto, las fases y objetivos que se han ido superando, los problemas que han ido surgiendo a lo largo de éste y cómo se han ido solventando. Con esto daremos una idea más específica de la dificultad a la que puede llevar desarrollar una herramienta de este tipo.

2.-OBJETIVOS

En el presente proyecto hemos diseñado la estrategia de planificación para que un robot móvil Surveyor SRV-1 con visión estereoscópica alcance un objetivo a través de una trayectoria que evita los obstáculos del escenario. Para ello se han desarrollado los siguientes módulos:

- Módulo de comunicación del Computador con el SRV-1 a través de Wi-Fi.
- Módulo de simulación 3D para facilitar el diseño.
- Módulo de diseño de la trayectoria,
- Módulo de integración de un reconocedor de objetos mediante una dll externa.
- Módulo de posicionamiento de un objeto en un entorno tridimensional.
- Módulo de conexión con un sistema GPS a través de UDP.

3.-ESTADO DEL ARTE

3.1-Evolución de la robótica móvil

Los primeros robots móviles aparecen a mediados del siglo XX. William Grey Walter construyó dos robots móviles autónomos, Elmer y Elsie, en 1949. Entre 1961 y 1963, Beast se desarrolla en la Universidad Johns Hopkins. Beast es capaz de identificar tomas de corriente y aproximarse a ellas para recargar su batería de forma autónoma. A finales de los años 60, Shakey es desarrollado en el Instituto de Investigaciones de Stanford. Shakey es el primer robot móvil capaz de razonar sobre sus propias acciones. Mientras otros robots tenían que ser dirigidos mediante acciones simples para realizar una tarea compleja, Shakey analizaba la tarea compleja y la reducía a una secuencia de acciones simples A partir de este momento, la investigación, diseño y construcción de robots móviles creció de manera exponencial.

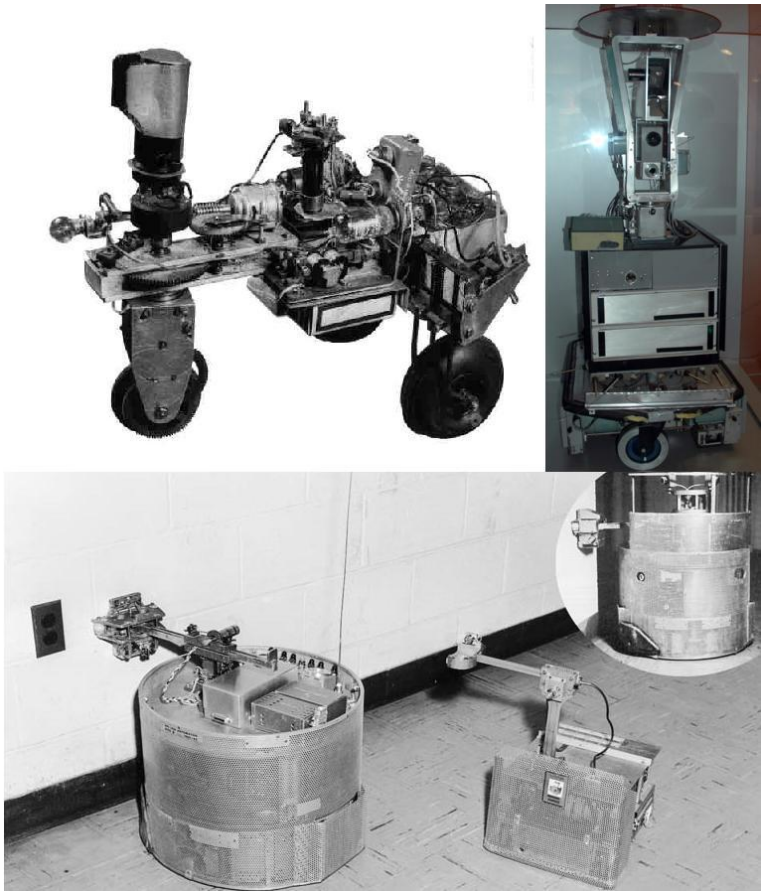


Ilustración 3: Primeros robots móviles. En la parte izquierda superior, Elmer de W. Grey Walter. En la parte derecha superior, Shakey. En la parte inferior, Beast.

En 1986 la empresa de origen japonés Honda creó el prototipo E0. Se tratada de un robot bípedo capaz de andar despacio en línea recta. Necesitaba aproximadamente 5 segundos para dar un paso. Al E0 le siguieron nuevas generaciones de prototipos que iban introduciendo nuevas mejoras. En 1993 el prototipo E6 caminaba de forma autónoma a velocidad humana y sorteaba obstáculos simples, pero carecía de tronco y extremidades superiores. Estos elementos serían incorporados en la siguiente generación, en la que se desarrollaron los prototipos P1, P2 y P3. En el año 2000 Honda presentaba la primera versión de ASIMO. ASIMO era más ligero y flexible que los prototipos, mejoraba los movimientos y tenía un aspecto más amigable. Cinco años después aparecía la segunda versión. Este modelo, entre otras mejoras, es capaz de desplazarse corriendo.

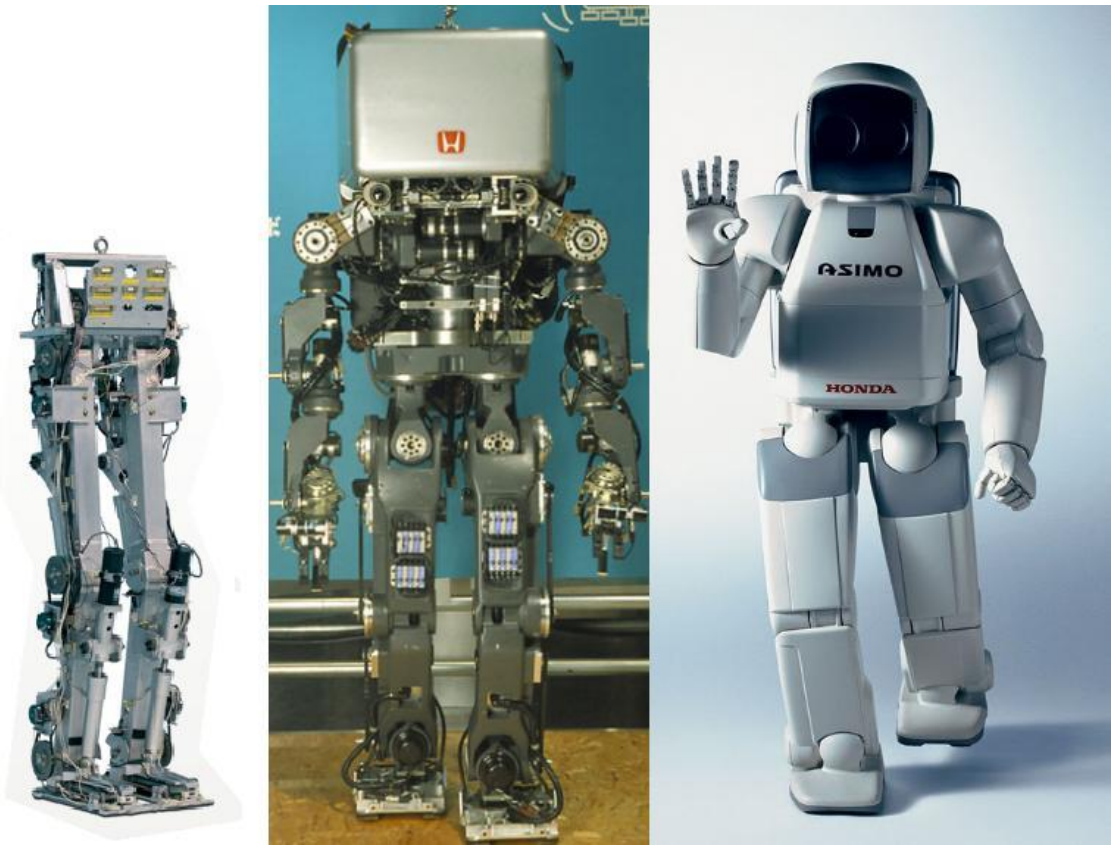


Ilustración 4: Evolución de los robots humanoides de Honda. De izquierda a derecha: E0, P1 y ASIMO v2.

Los robots móviles han sido satisfactoriamente utilizados con fines prácticos, por ejemplo, en la exploración de otros planetas. El 4 de julio de 1997 el rover Sojourner se convirtió en el primer robot móvil en llegar a la superficie de Marte. Era controlado por control remoto desde la Tierra, aunque incluía navegación autónoma usando un laser para detectar la presencia de obstáculos. En 2004 llegaron al Planeta Rojo los rovers gemelos Spirit y Opportunity dentro de la misión de exploración Mars Exploration Rover. Pertenecen a un modelo tecnológicamente más avanzado que el Sojourner. En este nuevo modelo se ha mejorado cualitativamente la

navegación gracias a la incorporación de un sistema de visión estereoscópica para el reconocimiento de obstáculos. Ambos rovers siguen en funcionamiento sobre la superficie de Marte a día de hoy.

Ilustración 5: Rovers utilizados en la exploración de Marte. A la izquierda, el Sojourner. A la derecha, el Spirit.

En 1999 Sony lanzó al mercado AIBO. Se trata de un robot mascota con apariencia de perro. Dotado de un gran número de sensores, AIBO es capaz percibir estímulos externos provenientes del entorno o de su dueño y actuar en consecuencia. Destaca su capacidad de aprendizaje.

Ilustración 6: Tres modelos AIBO de Sony. De izquierda a derecha: ERS-110 (1ª generación, 1999), ERS-210 (2ª generación, 2000) y ERS-7 (tercera generación, 2003).

En los últimos años se han empezado a comercializar robots diseñados para realizar tareas domésticas de forma autónoma. Entre ellos, podemos nombrar el robot limpiador Roomba de iRobot y el cortador de césped Automower de Husqvarna.

Ilustración 7: Robots domésticos. A la izquierda, Roomba. A la derecha, Automower.

3.2-La visión artificial

Uno de los propósitos de la visión artificial es lograr que un computador o robot "entienda" una escena o las características de una imagen y que este entendimiento le sea útil para llevar a cabo una misión. La visión estereoscópica es una de las técnicas de visión artificial donde existe más de una vista de una escena, tomada desde diferentes puntos. A través de estas imágenes se extraen las características tridimensionales de la escena y se hace el estudio pertinente a la aplicación en este caso gracias a la obtención de mapa de disparidades y el posterior tratamiento basándonos en colores.

De manera natural nuestro mecanismo de visión es estéreo, es decir, somos capaces de apreciar, a través de la visión binocular, las diferentes distancias y volúmenes en el entorno que nos rodea. Nuestros ojos, debido a su separación, obtienen dos imágenes con pequeñas diferencias entre ellas, a lo que denominamos disparidad. Nuestro cerebro procesa las diferencias entre ambas imágenes y las interpreta de forma que percibimos la sensación de profundidad, lejanía o cercanía de los objetos que nos rodean. Este proceso se denomina estereopsis.

En la estereopsis intervienen diversos mecanismos. Cuando observamos objetos muy lejanos, los ejes ópticos de nuestros ojos son paralelos. Cuando observamos un objeto cercano, nuestros ojos giran para que los ejes ópticos estén alineados sobre él, es decir, convergen. A su vez se produce la acomodación o enfoque para ver nítidamente el objeto. Este proceso conjunto se llama fusión.

Un factor que interviene directamente en esta capacidad es la separación interocular. A mayor separación entre los ojos, mayor es la distancia a la que apreciamos el efecto de relieve. Esto se aplica por ejemplo en los prismáticos, en los que, mediante prismas, se consigue una separación interocular efectiva mayor que la normal, con lo que se consigue apreciar en relieve objetos distantes que en condiciones normales no seríamos capaces de separar del entorno. Ésta técnica es aplicada en la fotografía aérea, en la que se obtienen pares estereoscópicos con separaciones de cientos de metros y en los que es posible apreciar claramente el relieve del terreno, lo que con la visión normal y desde gran altura sería imposible.

Una de las aplicaciones prácticas más antigua es la visualización y medición del relieve terrestre mediante fotografías aéreas. Si un avión toma dos fotografías de una zona de terreno con una cierta distancia calculada entre ellas, se obtiene un estéreo-par, que posteriormente puede verse en relieve con un estereoscopio especial. Si las tomas se realizan con la adecuada precisión, permiten calcular elevaciones en el terreno, para lo cual se emplean los estéreo-comparadores. A partir de datos del terreno pueden también generarse imágenes 3D simuladas mediante software específico.

De forma similar a la fotografía aérea, NASA ha obtenido numerosas vistas tridimensionales de fotografías de la Tierra obtenidas desde satélites, (ver http://www.jpl.nasa.gov) así como también de otros planetas de nuestro Sistema Solar. Las extraordinarias imágenes estéreo de la superficie de Marte obtenidas por la sonda Pathfinder de la NASA son otro ejemplo de aplicaciones para el estudio de otros planetas. La toma de imágenes en estéreo no solo sirvió para ver la superficie de Marte en 3D, sino para calcular distancias y tamaños de las rocas y conducir con más seguridad el vehículo.

Los sistemas de video-cámaras estéreo permiten operar en entornos peligrosos u hostiles con la máxima precisión.

3.3-Navegación de un robot.

La navegación es la ciencia de conducir un robot móvil mientras atraviesa un entorno para alcanzar un destino o meta sin chocar con ningún obstáculo.

La navegación nos lleva a crear una series de herramientas totalmente necesarias para la creación de una trayectoria viable y segura. Hemos de crear un mapa con la información que tenemos o ir creándolo según exploremos. Para ello tenemos que tener en cuenta varias variables importantes como son la percepción del entorno, fusión de sensores, control de movimiento…

Planificar es prever y decidir hoy las acciones que nos pueden llevar desde el presente hasta un futuro deseable, no se trata de hacer predicciones acerca del futuro sino de tomar las decisiones pertinentes para que ese futuro ocurra.

La idea general en la planificación de trayectorias consiste en desvincular el problema de la cinemática y dinámica del robot del de la obtención de una ruta (óptima o no) libre de obstáculos.

De todos los métodos que se pueden emplear el más recomendable en nuestro caso por rapidez, tipos de sensores que se van a emplear, topología del terreno y demás es un método de ocupación de celdas fijas. Básicamente, se divide el espacio de trabajo en celdas de determinada área.

En este caso nos hemos basado en el algoritmo A* para hacer la búsqueda de la trayectoria a través del grafo. Escogimos éste por ser eficiente y flexible, aparte de tener como propiedad ser un algoritmo completo, es decir, si existe una solución siempre dará con ella. Esto es importante puesto que nos da fiabilidad y estabilidad a la hora de calcular una trayectoria.

4.-ENFOQUE Y METODOLOGÍA

Se nos propuso desde un primer momento un proyecto dinámico y cambiante que además iba a depender de otros equipos de trabajo encargados de la elaboración de otros módulos que serían vitales para el proyecto, por lo que entendimos que la necesidad de colaborar con tantas unidades de trabajo hacía pertinente seleccionar un enfoque que nos proporcionase una metodología lo suficientemente flexible para poder acabar satisfactoriamente el proyecto y a su debido tiempo. Por ello consideramos conveniente enfocar el proyecto desde el paradigma de desarrollo ágil de software que entiende que el cliente es también parte activa del proceso, lo que nos permite ir incorporando todas las propuestas del cliente y solventándolas en poco tiempo.

Entendimos que debíamos considerar a nuestro coordinador como cliente al que debíamos satisfacer, puesto que desde un primer momento se nos había comunicado que el proyecto se nos iría modificando a medida que avanzásemos.

Como ya hemos mencionado en diferentes apartados decidimos utilizar para el desarrollo de este proyecto una metodología OpenUP. Según Pressman (2005) la metodología Open Unified Process, o también conocida en castellano como Proceso Unificado, entiende el proyecto como un ciclo de vida del que los componentes del equipo son parte orgánica del mismo. Con lo cual la participación activa, el intercambio de ideas y el aprovechamiento de los conocimientos y diferencias del equipo permiten dar salida a cada problema surgido con relativa inmediatez, y produce una retroalimentación temprana en cada ciclo que contribuye al reciclaje en cada momento del proceso, mucho antes de ser presentado.

5.- DE DESARROLLO DEL PROYECTO

Fase inicial: familiarización con el entorno de desarrollo y selección de herramientas a utilizar

- Instalación del entorno de desarrollo y pruebas y ejemplos básicos del lenguaje de programación C#.
- Aprendizaje del funcionamiento del motor gráfico Truevision 3D 6.5.
- Pruebas del resto de herramientas necesarias para la realización del proyectos, como por ejemplo la creación de un repositorio en Google Code o el diseño de los elementos 3D en herramientas de modelado (por ejemplo: AutoDesk Maya).

Fase secundaria:

- Diseño de módulos necesarios para la realización del simulador
- Primeros esbozos de la interfaz
- Pruebas con el motor de física Newton integrado dentro del motor gráfico TrueVision 3D 6.5.
- Modelado gráfico del SRV-1 y mapeado del entorno grafico.
- Ajuste de la física y movimientos del SRV-1 dentro del motor gráfico
    - Aquí surgió el problema con el ajuste de la física del motor Newton. Tenemos que ajustar peso, fuerzas en las ruedas, fricción de ruedas y terreno del simulador y ajuste del timer respecto a la física calculada de manera constante. Todas estas variables dependen las unas de las otras y un ajuste óptimo nos da como resultado un movimiento constante y preciso en el simulador.
    - Véase anexo Timer.

Fase tercera:

- Desarrollo del modulo de planificación de caminos
    - Solventar problemas de colisión considerando distancias de seguridad respecto a todos los objetos del entorno de simulación.

- Una vez terminado el simulador comenzamos con la integración de las librerías Aforge para el manejo del robot.
- Empezamos con el tratamiento de imágenes basándonos en colores.
    - Anexo tratamiento de imágenes con métodos unsafe.
- Se desarrolla la formula trigonométrica para situar el objeto en el entorno 3D.
    - Véase anexo posicionamiento de un objeto en entorno 3D
- Se desarrolla el modulo GPS con múltiples hilos de ejecución.
- Se realiza el modulo parser xml y se realizan todas las pruebas necesarias
- Ajuste de movimientos del robot SRV-1.
    - Problemas con las librerías Aforge debido a ciertas limitaciones, fue necesario recompilarlas.
- Desarrollo de ajustes del entorno real al simulador.
    - Las distancias respecto al simulador y la realidad son distintas, para ello tuvimos que conseguir una constante que nos relacionase distancias del simulador-realidad y de realidad-simulador.

Fase cuarta:

- Pruebas de sincronización y tratamiento de imágenes con el simulador y el robot SRV-1.
    - Comprobación del posicionamiento 3D. Funciona correctamente.
- Terminamos el modo simulador y realizamos el modo de ejecución Surveyor y Ambos.
    - Avance en el proyecto: el objeto se localiza, se crea en el escenario del simulador, podemos planificar y enviar las acciones necesarias para evitar un obstáculo: OBJETIVO PRINCIPAL DEL PROYECTO.

Fase quinta:

- Realización del modo de ejecución Ambos con visión.
    - Avance en el proyecto: no nos limitamos a sólo captar el escenario con una foto, ahora haremos fotos en cada giro de la trayectoria

analizando si hay nuevos obstáculos, si los hay se ponen en nuestro simulador y si ya se habían detectado no se ponen de nuevo. Si se detectan nuevos objetos re-planificamos el camino a nuestra coordenada de destino.

- Integración del algoritmo de análisis de detección de objetos del otro grupo.
    - Incorporación de dll externa para tratamiento de imágenes.
- Pruebas finales y generación de documentación

Fase sexta

- Pleno desarrollo de la memoria.
- Pequeños retoques en el código.

6.-ARQUITECTURA HARDWARE DEL SISTEMA

Los elementos centrales de la arquitectura del sistema son un computador y un robot móvil. El computador es el encargado de ejecutar la aplicación de simulación de este robot y el cálculo de rutas para la planificación. El robot móvil es el Surveyor SRV-1 dotado un módulo adicional para el sistema de visión estereoscópica.

Las características más relevantes del robot móvil Surveyor SRV-1 se encuentran detalladas a continuación.

- El diseño está basado en un código abierto, lo que ofrece libre acceso a sus esquemas y a su código fuente a través de una licencia GPL.
- Es totalmente programable para un modo de operación autónoma.
- Puede ejecutar programas escritos en C que hayan sido previamente almacenados en su memoria flash.
- Puede ser controlado vía red inalámbrica (WiFi 802.11b/g) con un alcance de hasta 100 metros en espacios cerrados. Alcance que se extiende hasta 1000 metros en espacios abiertos.
- Puede ser controlado desde consola.
- El modelo SRV-1 utilizado incorpora para la visión estereoscópica dos módulos de cámaras SRV-1 Blackfin separados por 10.74 cm. Cada uno de estos módulos incluye:
    - Un procesador Blackfin BF537 a 500 MHz con 32 MB de memoria SDRAM y 4 MB de memoria flash.
    - Una cámara Omnivision OV9655 a 1.3 megapíxeles con una resolución máxima de 1280x1024 píxeles.
- Incorpora para su locomoción orugas tipo tanque.
- La velocidad del SRV-1 es de entre 20 cm/s y 40 cm/s.
- El peso es de 350 gramos.
- Las dimensiones del SRV-1 son reducidas: 120 mm de longitud x 100 mm de ancho x 80 mm de alto.

Los requerimientos mínimos del computador para poder ejecutar la herramienta de una manera fluida son:

- Procesador Core2 Duo 2 GHz o Amd Athlon.
- Tarjeta gráfica dedicada de 128 MB
- Tarjeta de conexión inalámbrica 802.11g
- Sistema Operativo Windows XP o superior.
- 1 Gbytes de memoria RAM
- .NET FRAMEWORK 2.0 o superior.

En la ilustración de la página siguiente se puede observar cómo interaccionan las distintas partes que componen este arquitectura.

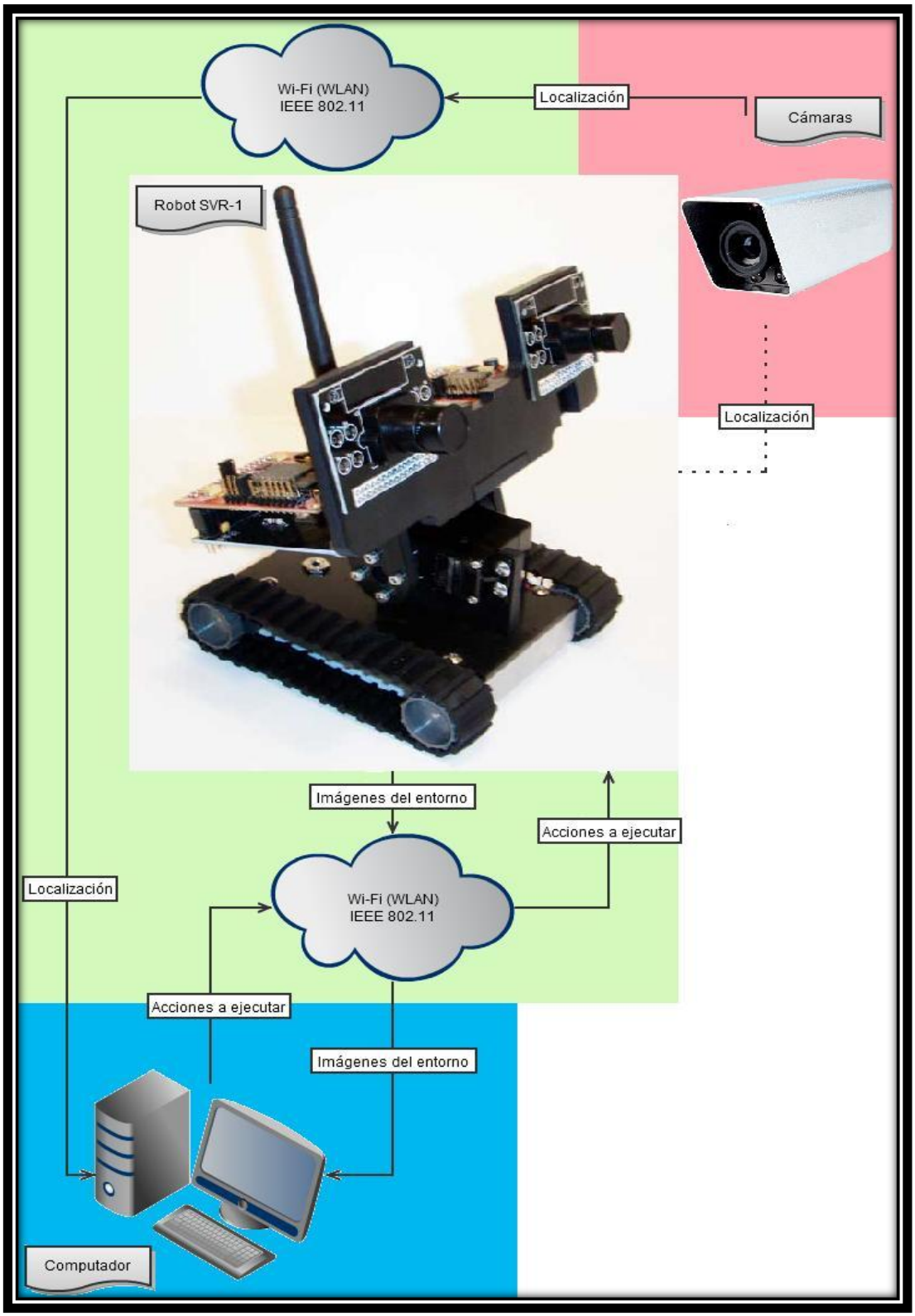


Ilustración 8: Esquema de la arquitectura hardware

7.-ARQUITECTURA SOFTWARE DEL SISTEMA

Hemos llevado a cabo una modularización del sistema tras la cual han resultado los siguientes apartados:

7.1.-Módulo grafico

Este módulo está compuesto principalmente por las clases de manejo del motor grafico, se realizo una separación con el objetivo de realizar un código reutilizable y con el que pudiésemos trabajar de manera independiente todos los miembros del grupo.

El resultado es el siguiente:

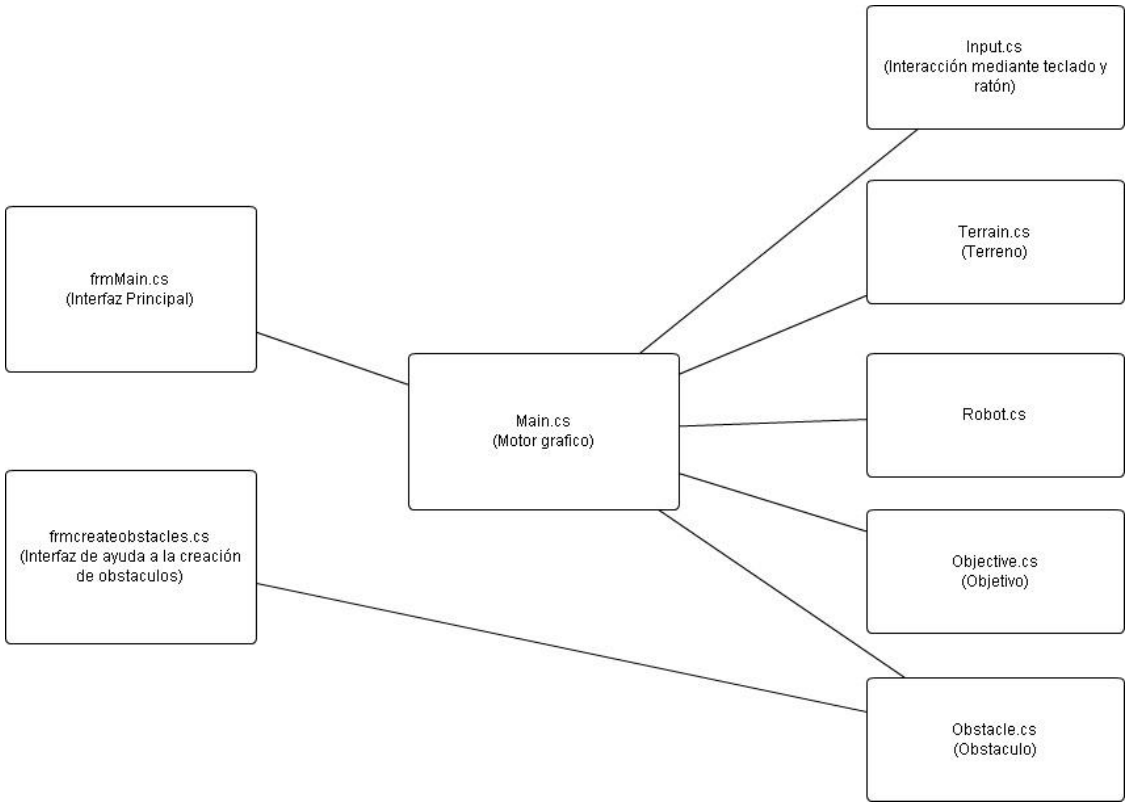


Ilustración 9: Visión general del módulo gráfico

Gracias a esta división de las clases se pueden realizar cambios en el código para una visualización distinta en pantalla de una manera rápida y aislar cualquier problema a un par de clases.

La clase Main.cs es la más complicada de todas porque incluye toda la carga grafica junto con funciones de ayuda para captura de imágenes en el simulador y la carga de texturas e inicialización del motor de física.

El resto de clases facilitan la creación y destrucción de los correspondientes objetos y la interacción con ellos excepto la clase Input.cs cuya única función es la de dar interactividad con el usuario por medio del teclado y el ratón.

7.2.-Módulo de planificación de trayectorias

El módulo de planificación de trayectorias se encarga de encontrar una ruta transitable entre una posición de origen y una posición objetivo. Para el cálculo de la trayectoria utilizamos el algoritmo A* por su eficiencia. En concreto, el algoritmo A* utilizado está incluido en el motor truevision3d.

Al algoritmo A* le debemos proporcionar información del entorno a través de una matriz de pesos (costes). Esta matriz se calcula en el mapeado a partir de la lista de obstáculos.

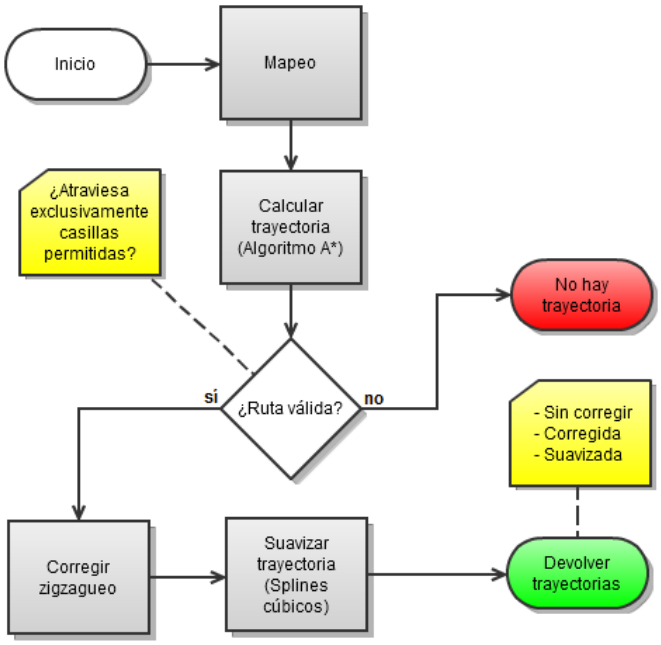


Ilustración 10: Planificación de trayectorias

El mapeado se lleva a cabo a partir de la lista de obstáculos. El primer paso, es crear una cuadrícula. Inicialmente, todas las casillas son permitidas. A continuación, se prohíben las casillas que contengan total o parcialmente un

obstáculo. Por último, se establecen como casillas de seguridad las contiguas a las prohibidas. El proceso se detalla en la siguiente ilustración:

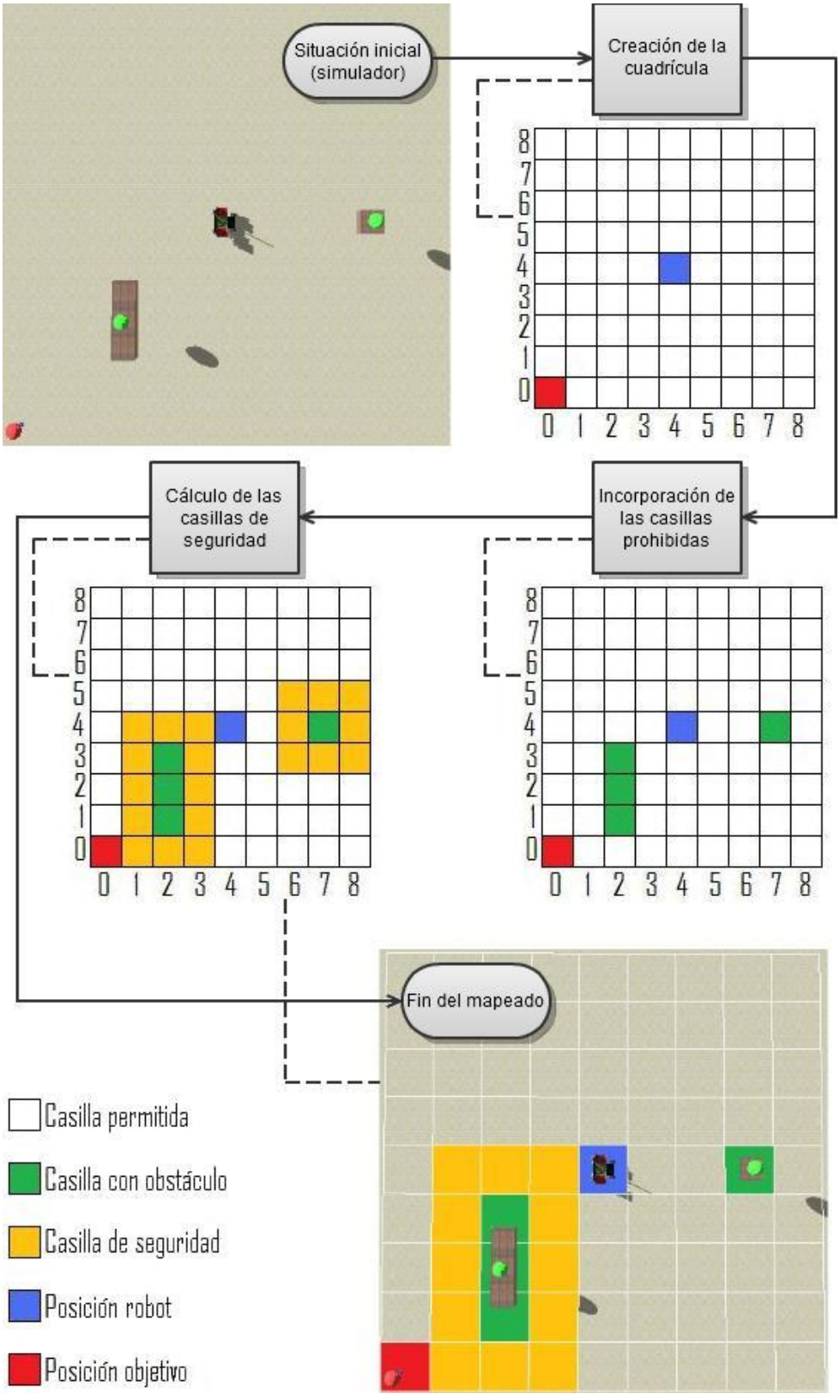


Ilustración 11: Resumen del mapeado

Como cabría esperar, los obstáculos lejanos no influyen en el mapeo. Por obstáculo lejano se considera cualquier obstáculo que no está localizado ni siquiera parcialmente en la cuadrícula.

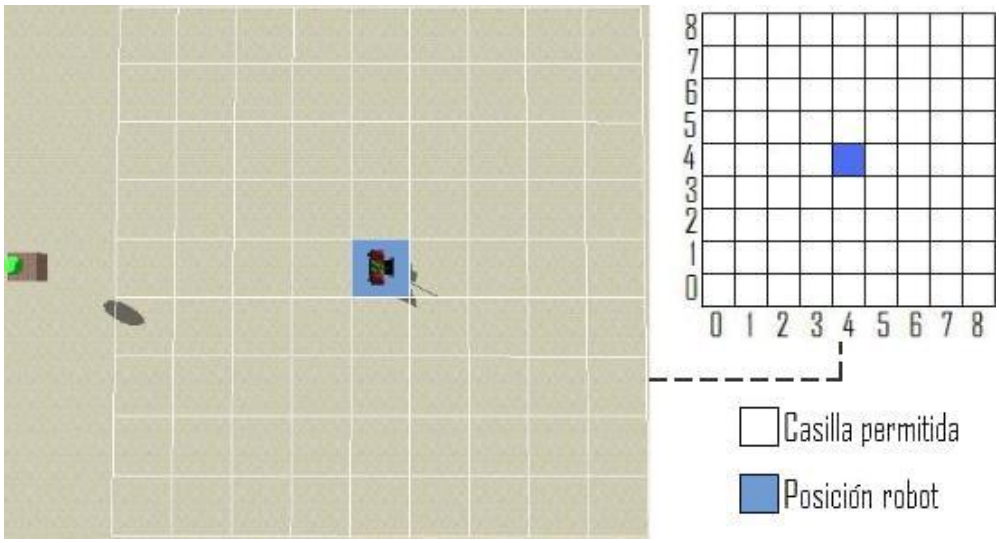


Ilustración 12: Mapeado con obstáculos fuera de la cuadrícula

La incidencia anterior, que haya obstáculos fuera de la cuadrícula, es habitual. Asimismo, aunque menos frecuente, el objetivo marcado podría exceder los límites de la cuadrícula. En ese caso, se tomará como objetivo real de la planificación aquella casilla que siendo permitida esté más cerca del objetivo marcado. Hay que recordar que las dimensiones de la cuadrícula no son constantes, esto es, se pueden cambiar. Razón por la cual, adaptando las dimensiones de la cuadrícula a las necesidades de la sesión de trabajo se evita esta situación. No obstante, ha sido prevista.

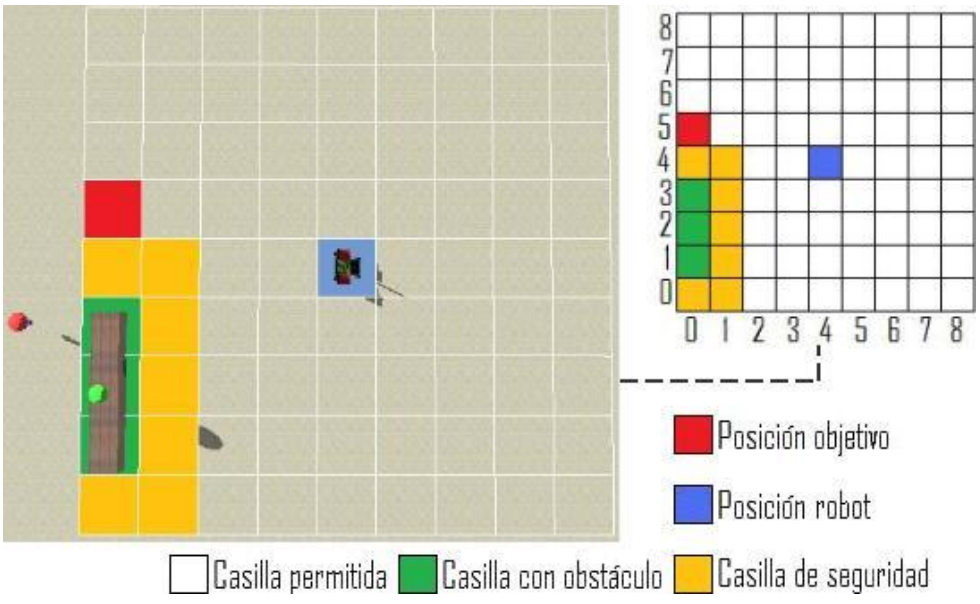


Ilustración 13: Mapeado con la posición objetivo fuera de la cuadrícula

Por último, apreciemos un último contratiempo: la finitud del escenario en la simulación 3D. El escenario es extenso, pero no ilimitado. Debemos, por consiguiente, impedir que el robot de la simulación se precipite al vacío. Consideramos, por tal razón, a toda casilla que excede los límites del escenario como casilla con obstáculo y, por tanto, dentro del grupo de las casillas prohibidas. Como en el caso general, para evitar riesgos innecesarios, estas casillas prohibidas estarán rodeadas por las correspondientes casillas de seguridad.

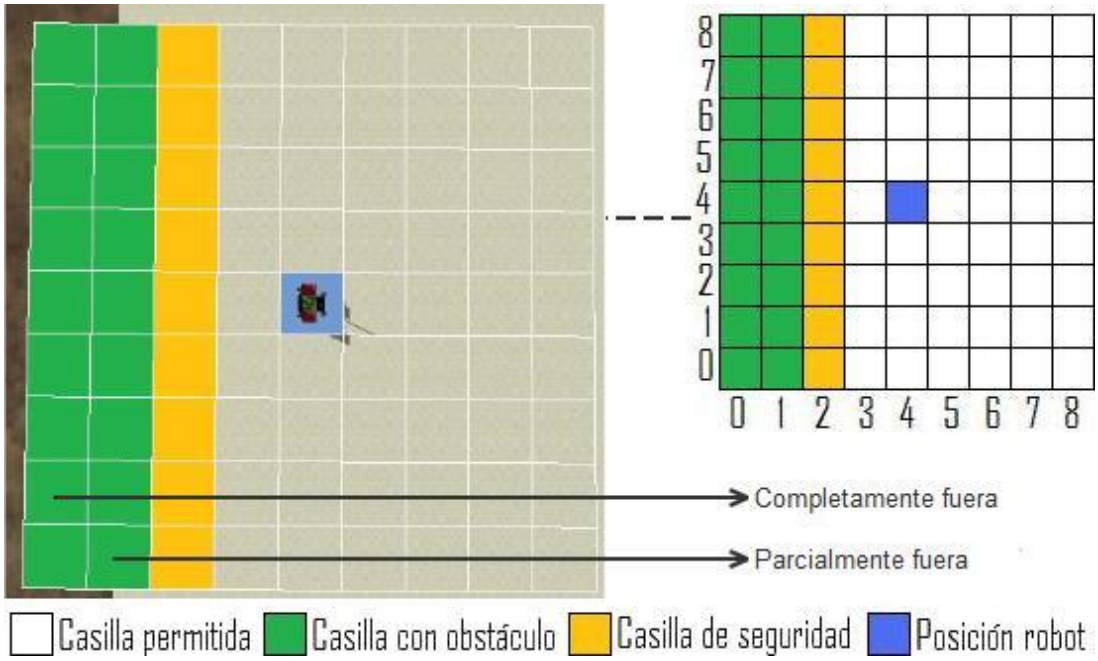


Ilustración 14: Mapeado con la cuadrícula parcialmente fuera del escenario

Finalizado el mapeado, se aplica el algoritmo A* a partir de la matriz de costes (pesos) obtenida en la fase de mapeado. La trayectoria devuelta por el algoritmo A* contendrá zigzagueos. Este efecto es indeseado pues conlleva que el robot realice más giros de los necesarios para recorrerla. Corregiremos, por tanto, la trayectoria quitando los zigzagueos.

El algoritmo que corrige la trayectoria es sencillo, se basa en eliminar todos los puntos intermedios posibles. Dados dos puntos A y B de una ruta, si la línea recta que los une atraviesa exclusivamente casillas permitidas, entonces los puntos intermedios pueden omitirse, de forma que queda A directamente unido con B en línea recta y, por tanto, sin zigzagueo.

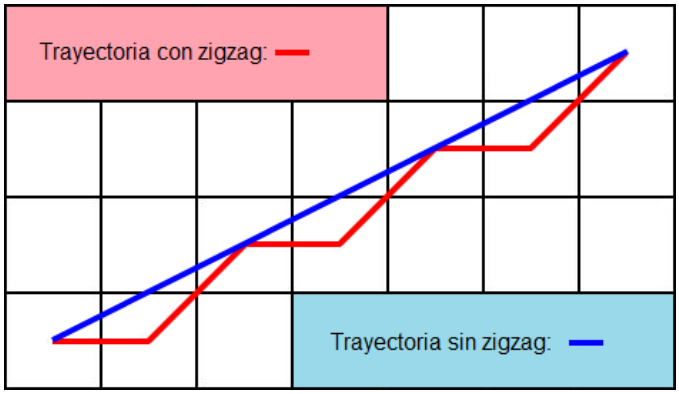


Ilustración 15: Corrección del zigzagueo de la trayectoria

Una vez corregida la trayectoria, se puede suavizar mediante splines cúbicos. El módulo de planificación aplica los splines cúbicos, aunque luego no se utiliza.

7.3.-Módulo del cálculo de acciones

Este módulo se encarga de obtener las acciones necesarias para la ejecución de la ruta por parte del robot. Estas acciones siguen un patrón establecido al que llamamos paso. En cada uno de los pasos el robot ha de realizar dos acciones esenciales: un giro y un avance. La totalidad de los pasos se agrupan a su vez en una estructura denominada lista de pasos.

En la implementación de este módulo aparecen tres clases: Listadepasos.cs, Paso.cs y Acción.cs. Listadepasos.cs es la utilizada desde el módulo principal. Su constructora recibe como parámetros la rotación del robot y la trayectoria.

En la ilustración de la siguiente página, aparece la estructura de la lista de pasos y su forma de ejecución.

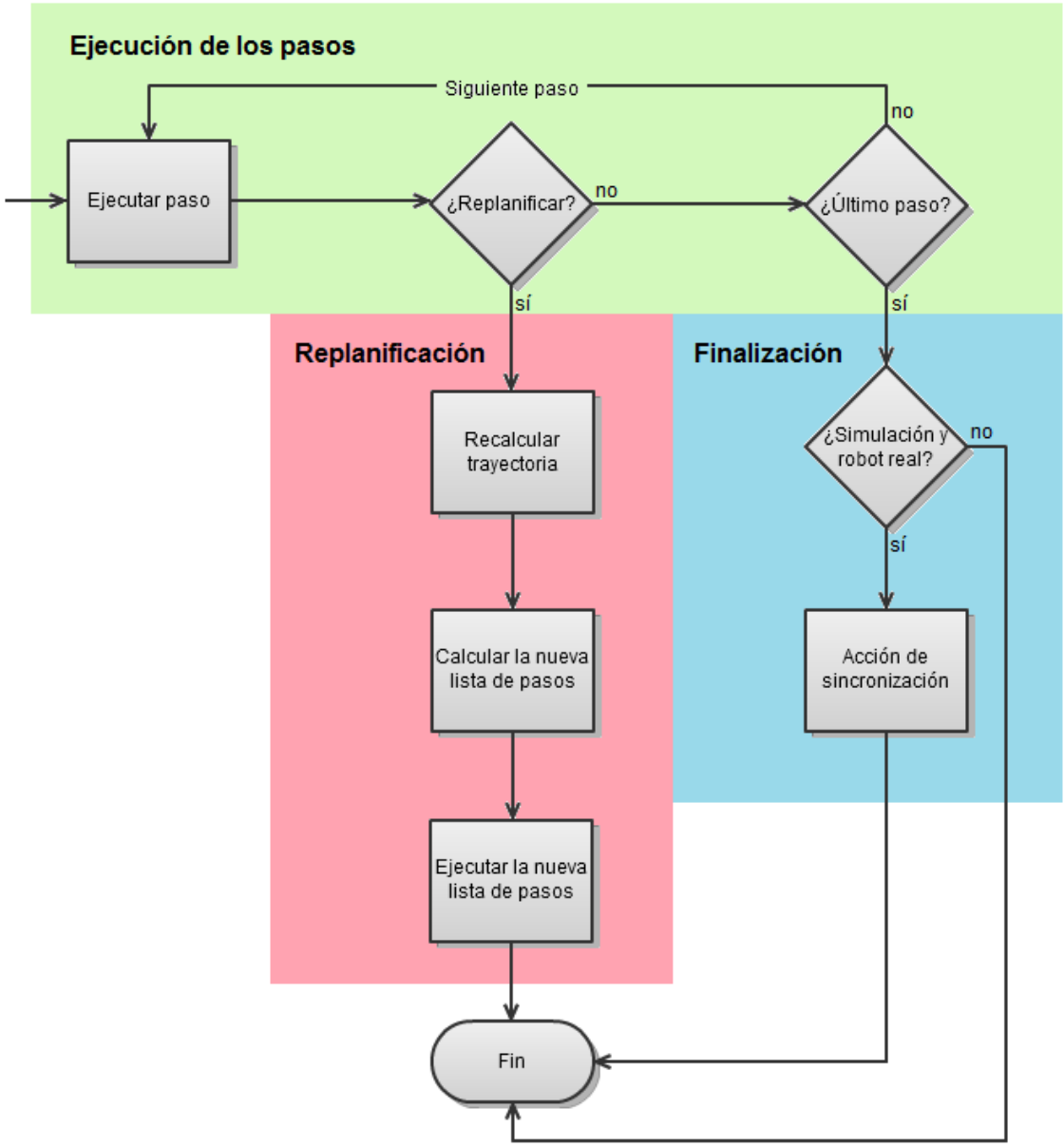


Ilustración 16: Lista de pasos

Como ya hemos comentado, las dos acciones principales de un paso son girar y avanzar. En una acción de giro el robot rota sobre su eje vertical el número requerido de grados. Este número jamás supera 180º puesto que disponemos de ambos giros: a derecha y a izquierda. Por su parte, en el avance, el robot se desplaza una cantidad señalada en la dirección y sentido indicados por su rotación actual.

Para que las acciones anteriores se traduzcan en la ejecución éxitosa de la ruta, son necesarias unas acciones complementarias que comprueben la posición del robot. En este tipo de acciones se compara lo que se esperaba a priori que realizara el robot con lo que realmente ha realizado. En concreto, hay dos acciones de comprobación. En una sólo se comprueba el giro, en la otra se comprueba además la posición. La primera se incluye tras el giro. La segunda, detrás del avance.

Las acciones de comprobación están presentes en todas las listas de pasos. Aunque, si el CPS no está disponible, se ignoran.

Se muestra en la siguiente ilustración la estructura de un paso y su ejecución.

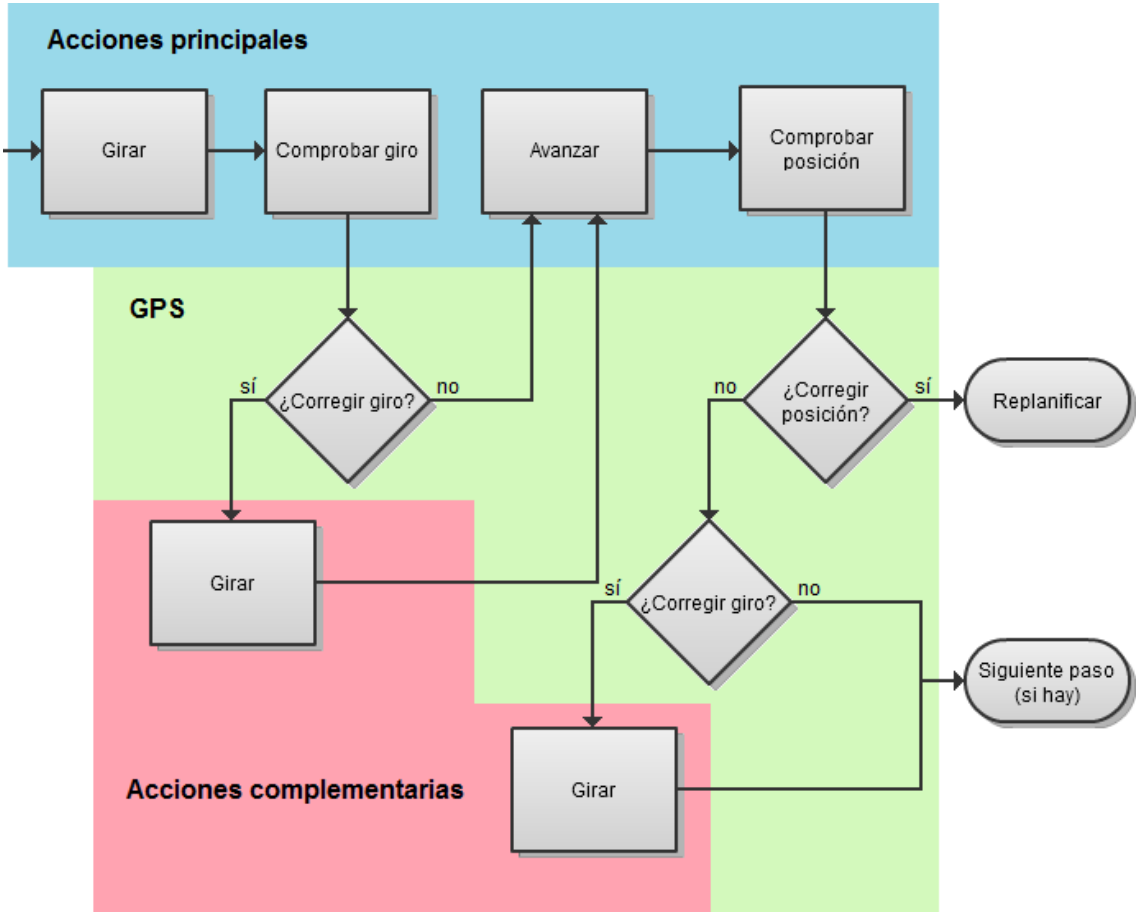


Ilustración 17: Paso

Además de las acciones comentadas hasta ahora, existen dos acciones más: visión y sincronismo. Estas acciones no están presentes en todas las listas de pasos.

La acción de visión estará presente si el usuario la ha pedido a través de la interfaz gráfica. En este caso, la acción de visión se añade en todos los pasos tras la comprobación de giro. Esta acción se encarga de capturar imágenes y comprobar si hay obstáculos nuevos. Si hay al menos un obstáculo nuevo, se vuelve a replanificar la ruta ya que se calculó sin tener en cuenta este obstáculo.

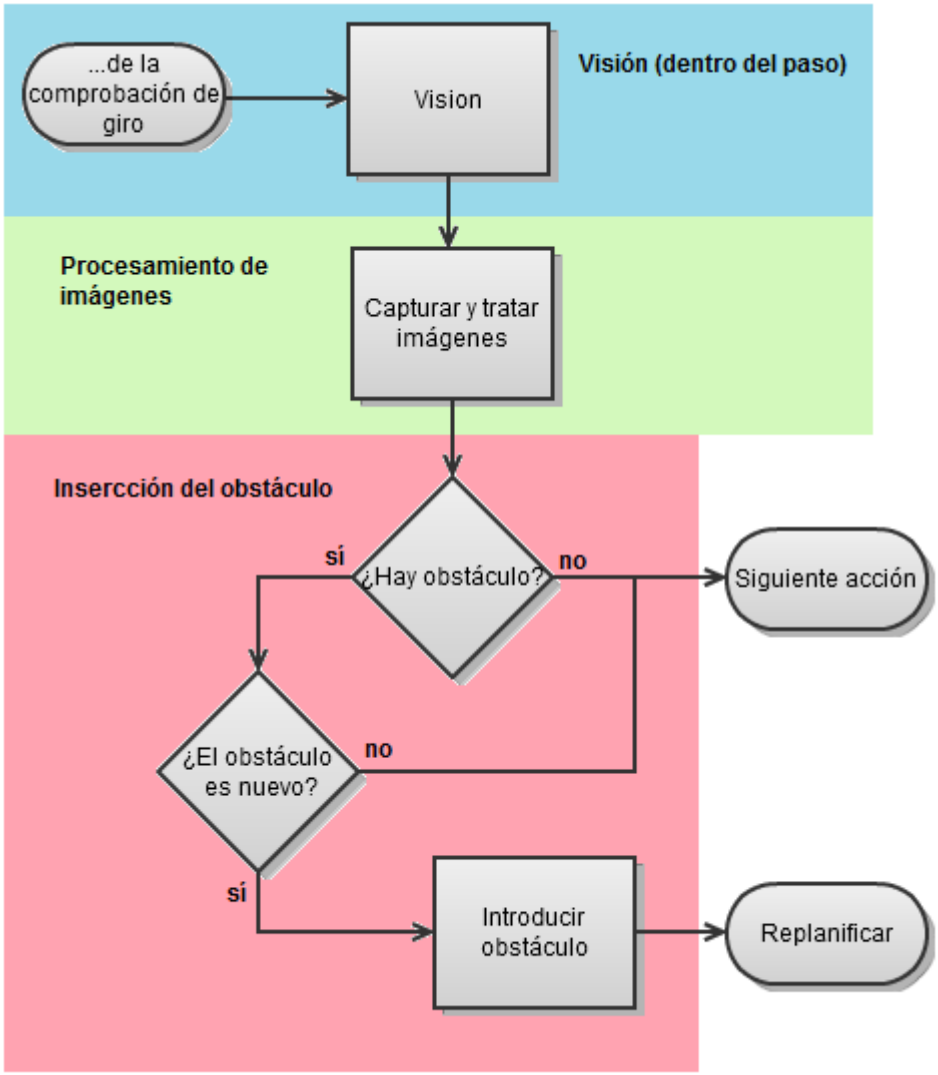


Ilustración 18: Acción de visión

La acción de sincronismo se introduce únicamente al final del último paso de la lista de pasos si ésta va a ser ejecutada simultáneamente por el robot real y el robot de la simulación. El propósito de esta acción es que el robot de la simulación tenga la misma rotación que el robot real al finalizar la ruta.

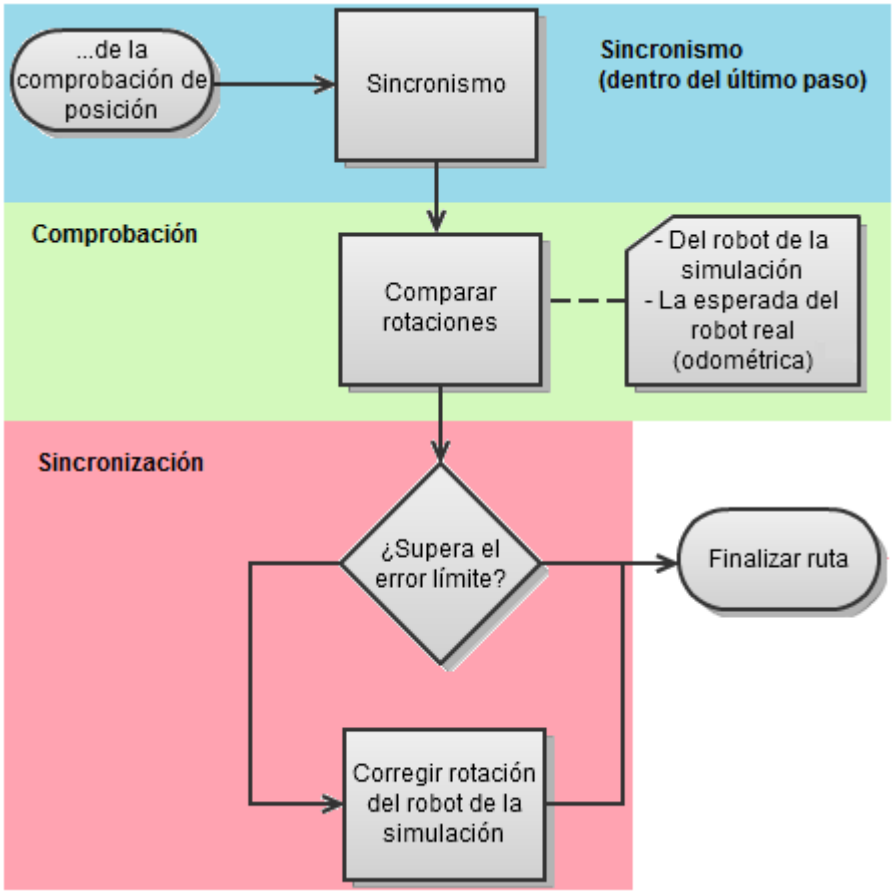


Ilustración 19: Acción de sincronismo

7.4.-Módulo búsqueda del objetivo

Este módulo se encarga de la ejecución del modo bola roja. En este modo de ejecución el robot ha de buscar una bola roja y aproximarse a ella.

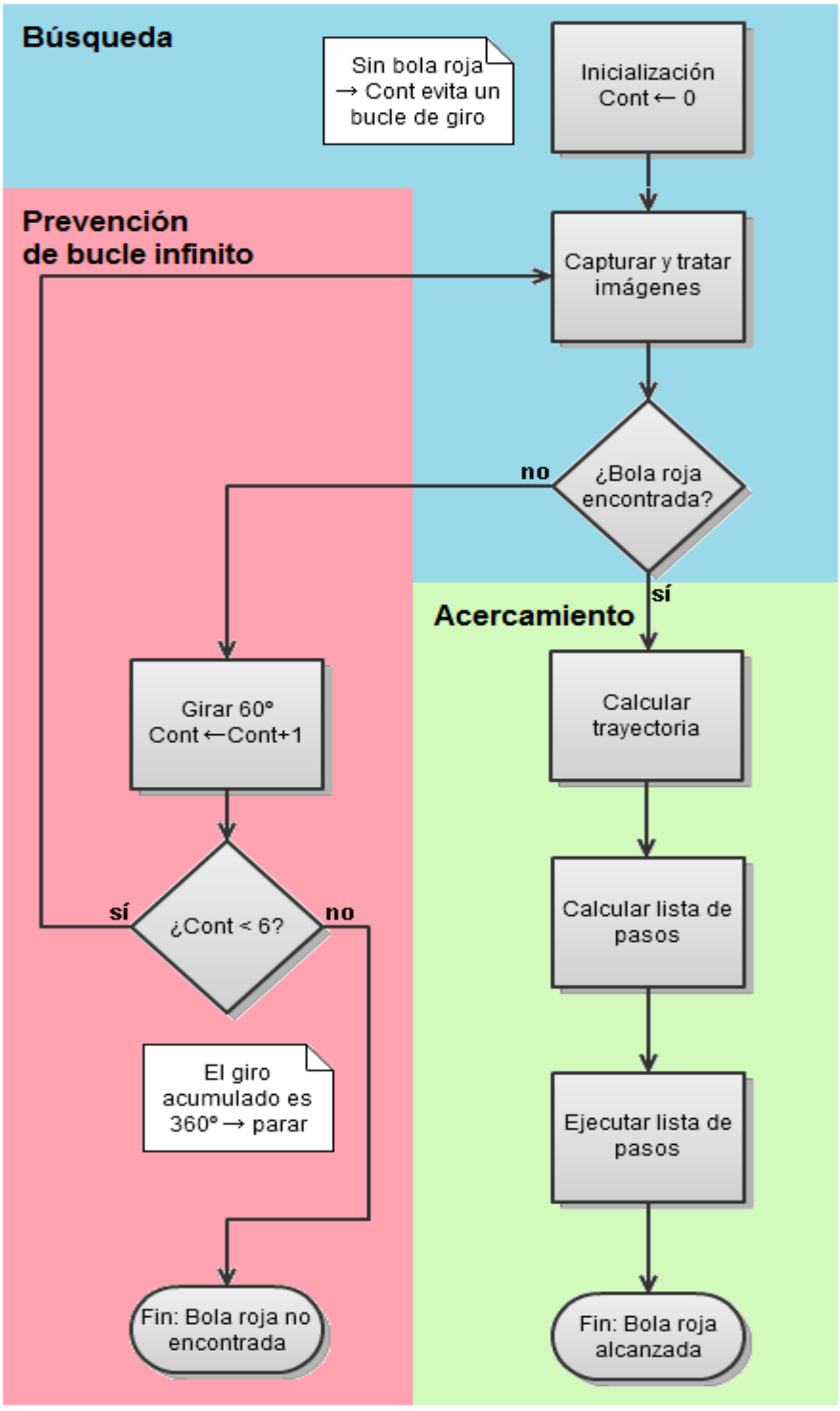


Ilustración 20: Busqueda de la bola roja

7.5.-Módulo de conexión con un sistema GPS a través de UDP

El módulo GPS se encarga del posicionamiento del robot a lo largo de la planificación y recorrido de las rutas.

Ilustración 21: Vista general de modulo de conexión GPS

La implementación de los módulos GPS o del posicionamiento mediante cámaras no se ha llevado a cabo pero si su preparación y pruebas dentro de la simulación. En nuestra simulación el robot comete errores al igual que los comete el robot de verdad debido al rozamiento aleatorio con la superficie y para ello añadimos estas posibilidades.

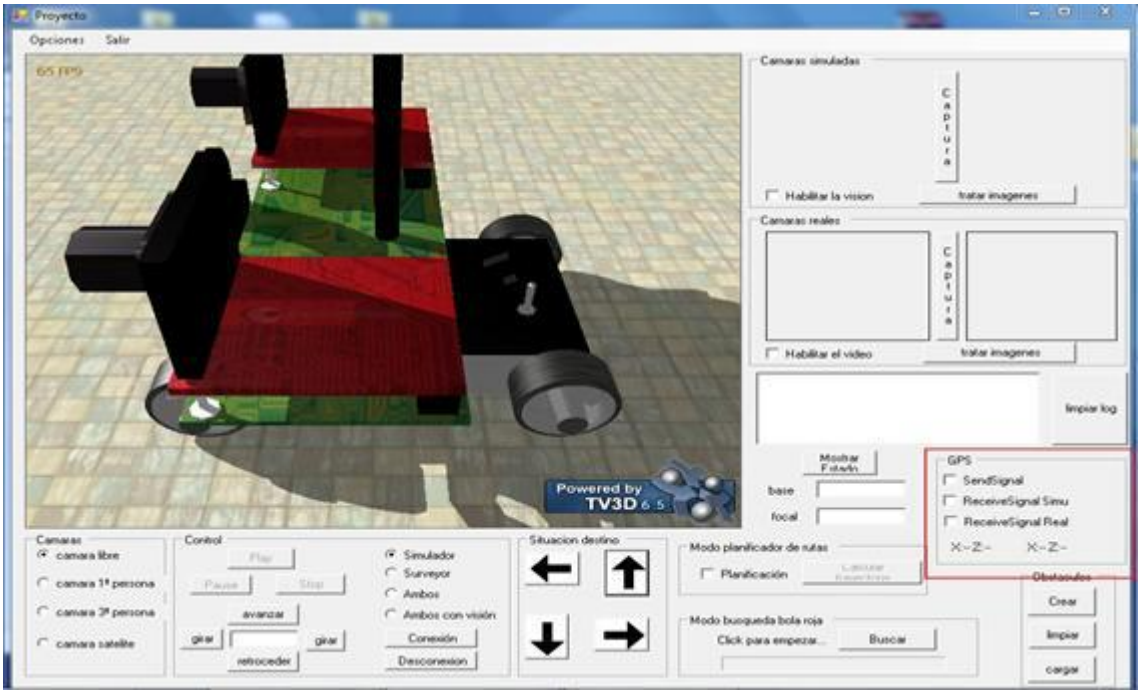


Ilustración 22: Modulo GPS dentro de la interfaz

En la interfaz grafica se encuentra un pequeño apartado donde activar el GPS, las dos primeras opciones sirven para emular el comportamiento en la simulación.

-SendSignal: Simulamos el envío de a través de una conexión UDP de las posiciones en las que se encuentra el robot, todo ello con hilos distintos al de la ejecución.

-ReceiveSignal Simu: Simulamos la recepción de una conexión UDP de las posiciones en las que se encuentra el robot, todo ello con hilos distintos al de la ejecución, en función de ello se corregirán los movimientos del robot.

-ReceiveSignal Real: Se deja preparado para poder recibir conexiones UDP que provengan de un emisor GPS o unas cámaras que realicen el posicionamiento.

Los dos indicadores de posición que se encuentran más abajo sirven como testigos de los datos que se envían cuando esta activada la simulación del envío y el testigo de la derecha para saber en todo momento los datos que se recibirían.

A continuación mostramos el diagrama de clases de los que está compuesto el módulo.

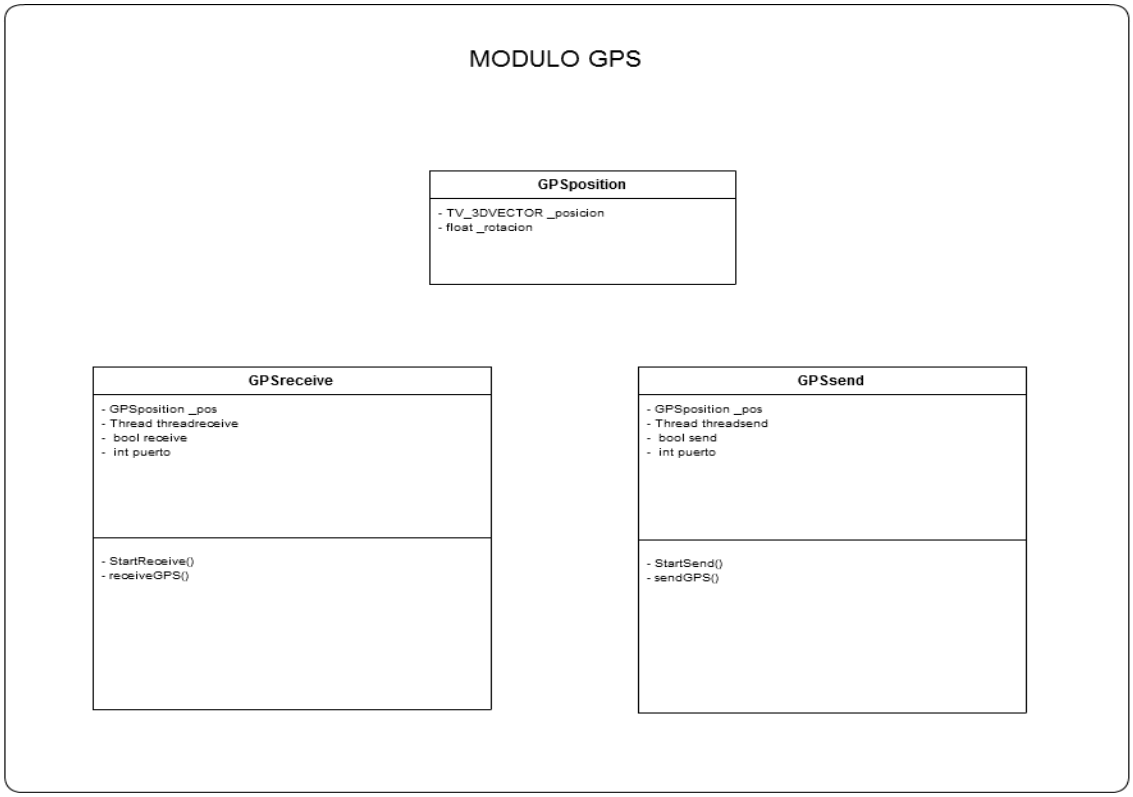


Ilustración 23: Diagrama de clases del módulo GPS

Se ha procurado realizar unas clases muy sencillas para que funcionando con hilos distintos sean fácilmente portables a cualquier aplicación. La inclusión de algún modulo verdadero de geoposicionamiento o posicionamiento incremental resultaría muy sencillo.

7.6.-Parser XML

El parser está preparado para tratar con dos tipos de documentos XML. Uno almacena los parámetros de configuración de la planificación de caminos. Otro contiene datos sobre cada uno los obstáculos presentes en el escenario.

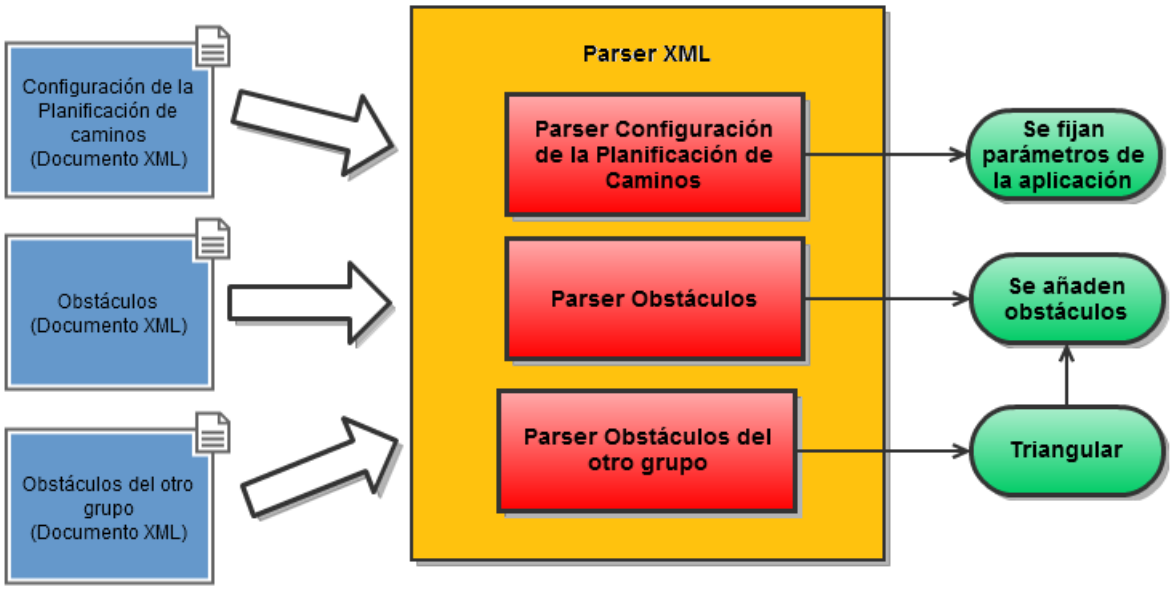


Ilustración 24: Parser XML

El documento XML de configuración de la planificación de caminos es parseado al inicio de la aplicación y configura los parámetros relacionados con la planificación de caminos. Se utilizan en el mapeado para la creación de la cuadrícula. Su estructura, así como los datos que guarda, se muestran a continuación.

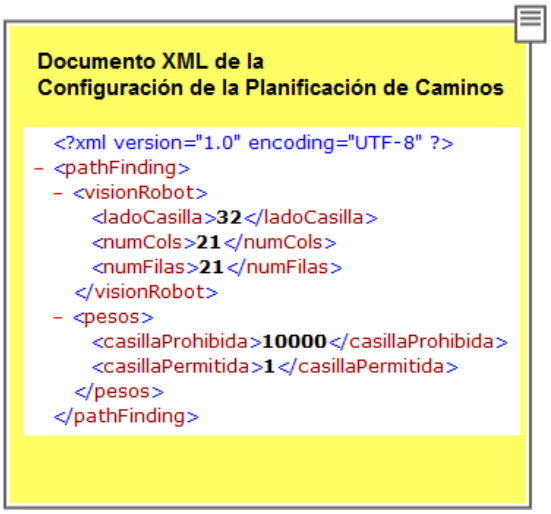


Ilustración 25: Estructura del documento XML de planificación

El documento XML de obstáculos es parseado cuando se lee un archivo XML. Los obstáculos que contiene son añadidos.

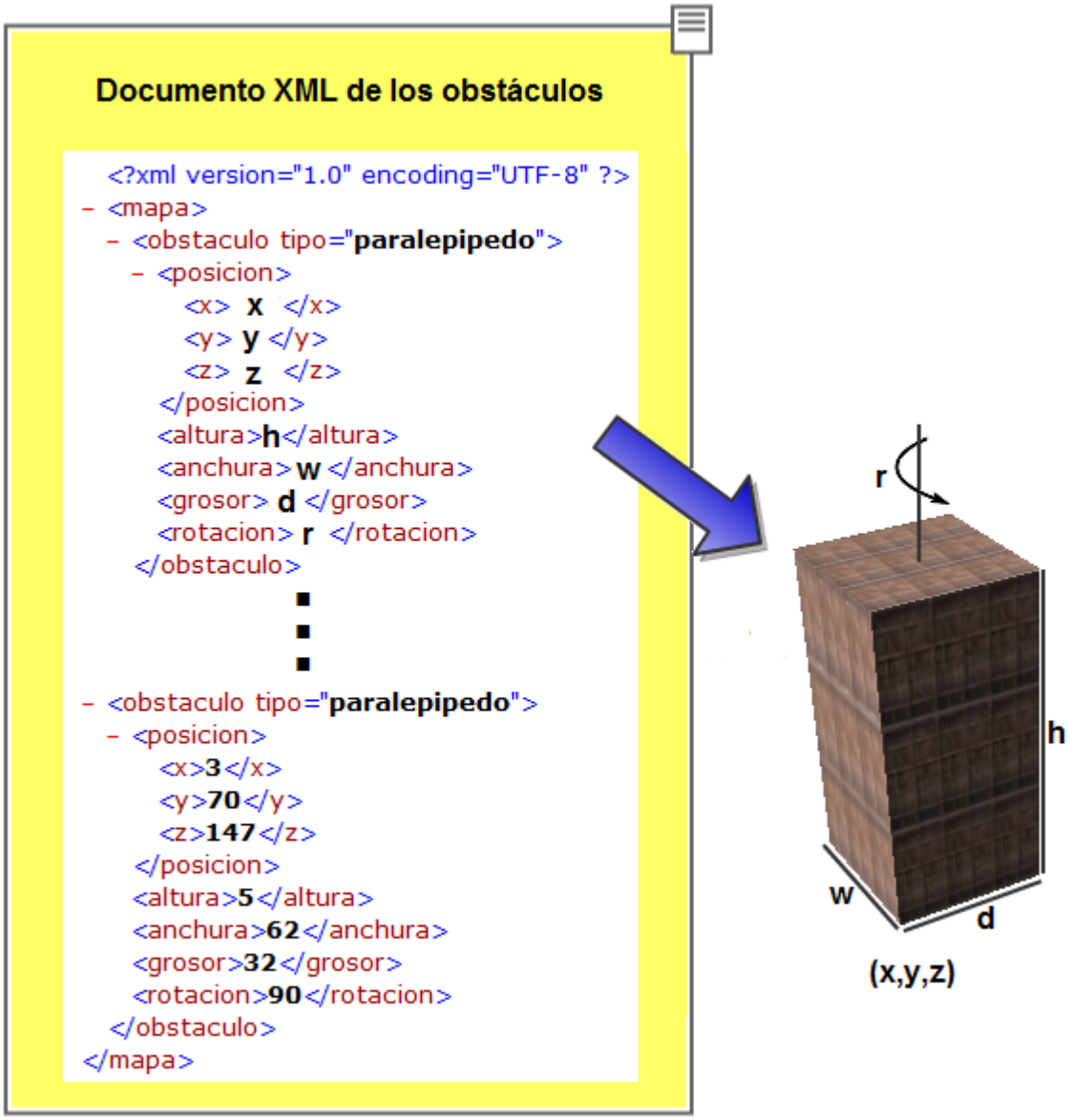


Ilustración 26: Estructura del documento XML de obstáculos

El documento XML de obstáculos del otro grupo es parseado cuando se encuentra un obstáculo al ejecutar una ruta en una acción de visión si se utiliza el algoritmo de detección de obstáculos del otro grupo.

Estos documentos XML siguen la estructura de los anteriores, pero el significado de sus campos es distinto. En la posición no viene la posición del objeto, sino que vienen los valores necesarios para calcularla, entre ellos, la disparidad.

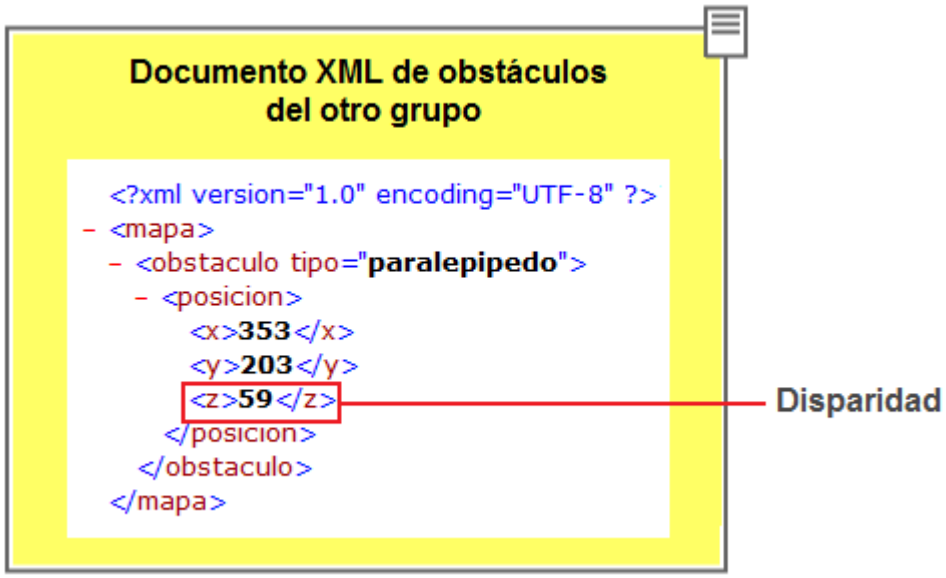


Ilustración 27: Estructura del documento XML de obstáculos del otro grupo

8.-HERRAMIENTAS UTILIZADAS

8.1.-Motor gráfico

El motor gráfico utilizado es el truevision3d versión 6.5, en sus comienzos sobre enero de 1999 empezó como un hobby desarrollado en Visual Basic 6 sobre directX 7 & 8 hasta el día de hoy que se ha convertido en producto comercial desarrollado íntegramente en C++ sobre directX 9 dando actualmente soporte para desarrollar en varios lenguajes C++, C#, Delphi and Visual Basic (6 y .NET).

El SDK incluye además del motor varias utilidades para ayudar al desarrollo de cualquier aplicación como un editor de shaders, un visor de modelos y plugins para exportar modelos desde las principales herramientas de modelado software (en nuestro caso Autodesk Maya).

TV3D está considerado como un motor 3D y no un motor para desarrollar juegos, su uso es gratuito mientras no te importe el logo de la empresa en la esquina inferior izquierda del render, si se desea quitarlo y realizar un producto comercial se necesita adquirir una licencia por un precio reducido comparado con otros motores.

La elección por este motor grafico fue fácil ya que podíamos desarrollar nuestro código con el lenguaje C# dentro del entorno de desarrollo visual studio también incluía un motor de físicas con una interfaz de manejo muy sencilla y además nos permitía utilizar modelos hechos con el programa de modelado que ya conocíamos (Autodesk maya), a todo ello hay que agradecer el soporte por parte de una comunidad bastante activa en los foros y en la wiki propia creada con ese fin

A grandes rasgos el motor gráfico ha sido elegido principalmente por estas características:

Sistema de render

- Modo ventana o pantalla complete con la posibilidad de intercambiarlos mientras se esta renderizando.
- Múltiples viewport en modo ventana.
- Múltiples modos de renderizado (por puntos, líneas o sólidos).
- Filtros de antialiasing y anisotrópico has 16x disponibles.

Sistema de terrenos y mapas

- Creación de terrenos sencilla gracias a la creación mediante mapas de altura.

Mallas

- Soporte completo de shaders.

- Soporte de animaciones a través de huesos y manipulación directa a través de llamadas internas.
- Herramientas de importación de los programas de modelación más importantes (blender, Max3d,Autodesk Maya).
- Detección precisa de colisiones.

Materiales y sistema de iluminación

- Soporte para luces de punto direccionales y foco.
- Gestión dinámica de incidencia de la luz sobre los distintos objetos.
- Soporte para bump mapping.
- Soporte de materiales con sus distintas propiedades.
- Soporte de mapas de luces para las mallas y el terreno.

Motor de Física Newton integrado

- Integración sencilla de todo tipo de objetos en el motor de físicas.
- Soporte completo para la integración de físicas de vehículos.

Mejoras adicionales

- Soporte completo UNICODE
- Incluye una librería matemática con todas las funciones estándar para vectores matrices y quaterniones

Estándares de la API

- Permite fácilmente la programación orientada a objetos.
- Fácil inicialización del motor.

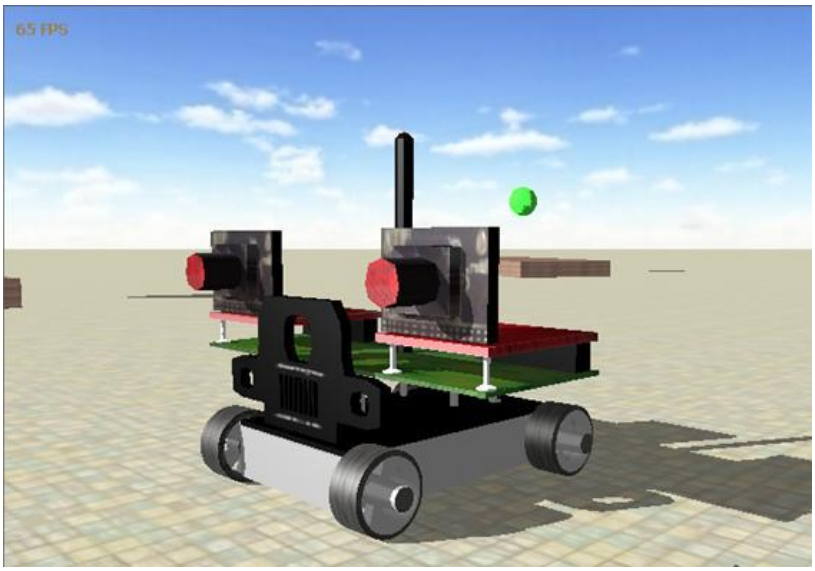


Ilustración 28: Primer plano del robot en el simulador 3D

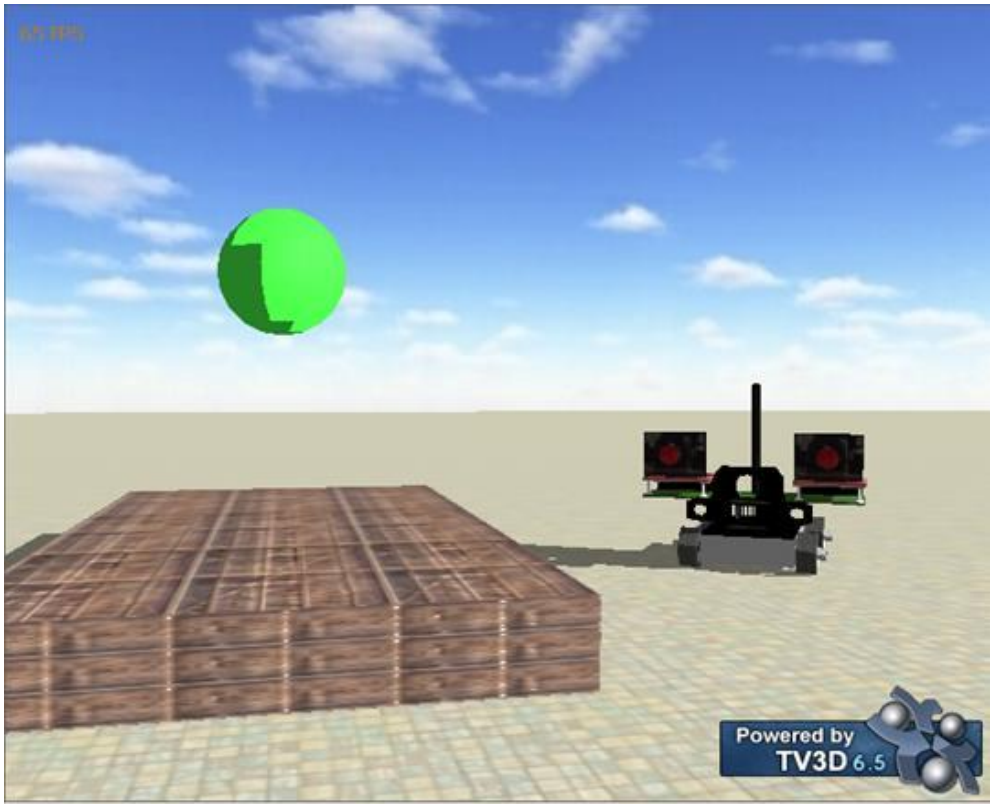


Ilustración 29: Captura del simulador 3D

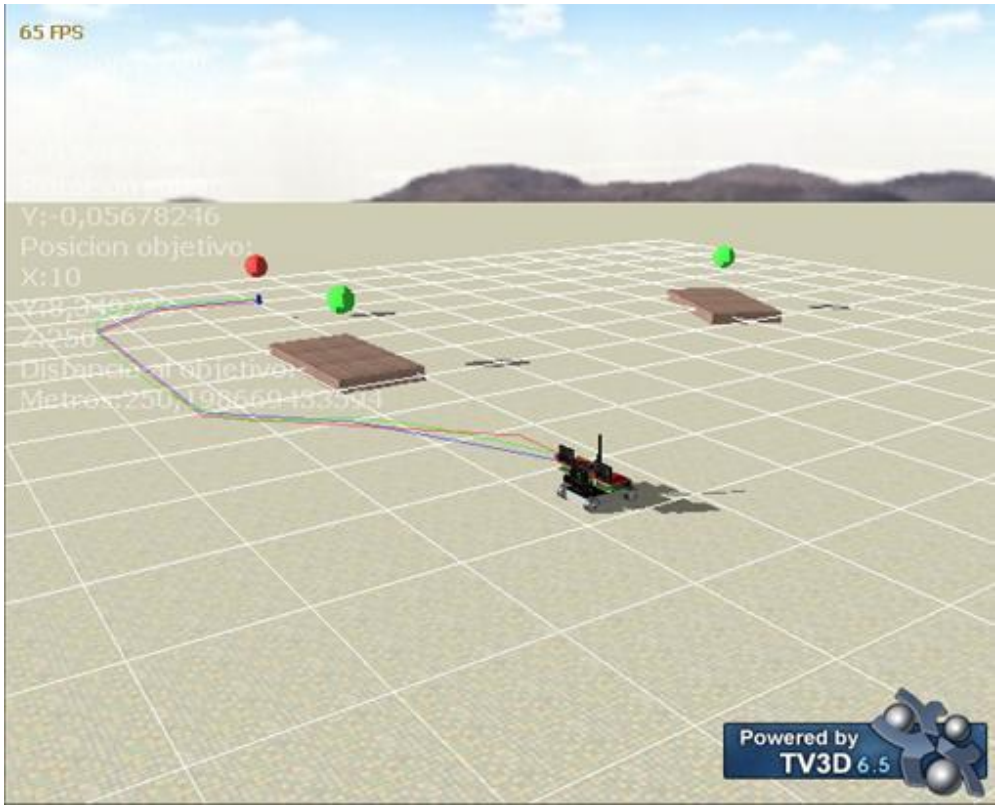


Ilustración 30: Captura del simulador 3D. Se aprecian las partes principales: robot, objetivo y obstáculos.

8.2.-Librería AForge

AForge.NET es un framework en C# de código abierto diseñada para desarrolladores en el campo de la visión por computador y la inteligencia artificial (procesamiento de imágenes, redes neuronales, algoritmos genéticos) en nuestro caso para el campo de la robótica y más concretamente nos ha permitido controlar el robot y todos sus sensores.

Hay que comentar que al ser de código abierto hemos tenido la posibilidad de recompilar la parte de la librería que nos interesaba para evitar un problema en la implementación de las mismas.

El problema consiste en una limitación de tiempo en el SRV-1 para ejecutar movimientos puesto que el envío vía WiFi es de un byte (256 valores) por tanto no se le puede mandar tiempos de avance de más de 2.55 s. La solución consistió en la modificación de estas librerías con un bucle que se repitiese el envío de 255 tantas veces como fuese necesario y el resto en otro envío, para ello utilizamos una división y un modulo de 255 respecto al tiempo de avance.

y era el hecho de que solo se podía mandar como mucho una instrucción de avance no mayor a una cantidad fija lo cual nos obligaba a descomponer un avance de 5 metros en varias instrucciones y observar como el robot se paraba cada cierto tiempo.

8.3.-Modelador

El software de modelado elegido ha sido Autodesk Maya, debido principalmente a que además de que los modelos obtenidos eran totalmente compatibles con la herramienta de conversión del motor grafico uno de los miembros del grupo conocía la herramienta y era capaz de modelar cualquier objeto.

Autodesk Maya (también conocido como Maya) es un programa informático dedicado al desarrollo de gráficos en 3d, efectos especiales y animación. Surgió a partir de la evolución de Power Animator y de la fusión de Alias y Wavefront, dos empresas canadienses dedicadas a los gráficos generados por ordenador. Más tarde Silicon Graphics (ahora SGI), el gigante informático, absorbió a Alias-Wavefront, que finalmente ha sido absorbida por Autodesk.

Maya se caracteriza por su potencia y las posibilidades de expansión y personalización de su interfaz y herramientas'. MEL (Maya Embedded Language) es el código que forma el núcleo de Maya, y gracias al cual se pueden crear scripts y personalizar el paquete.

El programa posee diversas herramientas para modelado, animación, render, simulación de ropa y cabello, dinámicas (simulación de fluidos), etc.

Tiene un uso muy extendido debido a su gran capacidad de ampliación y personalización.

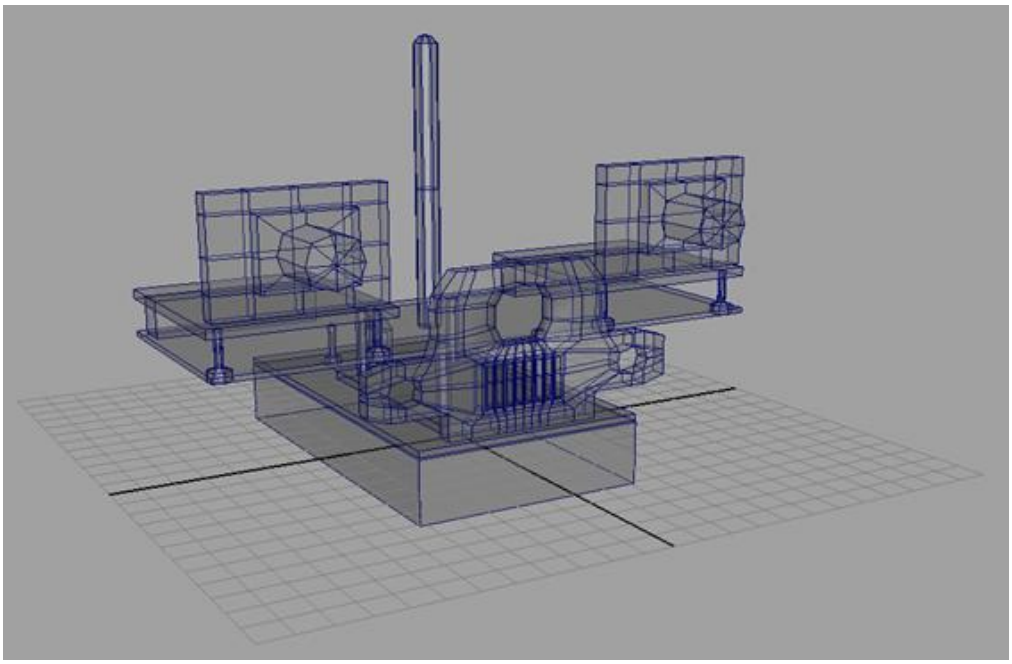


Ilustración 31: Captura de la malla básica en el modelador

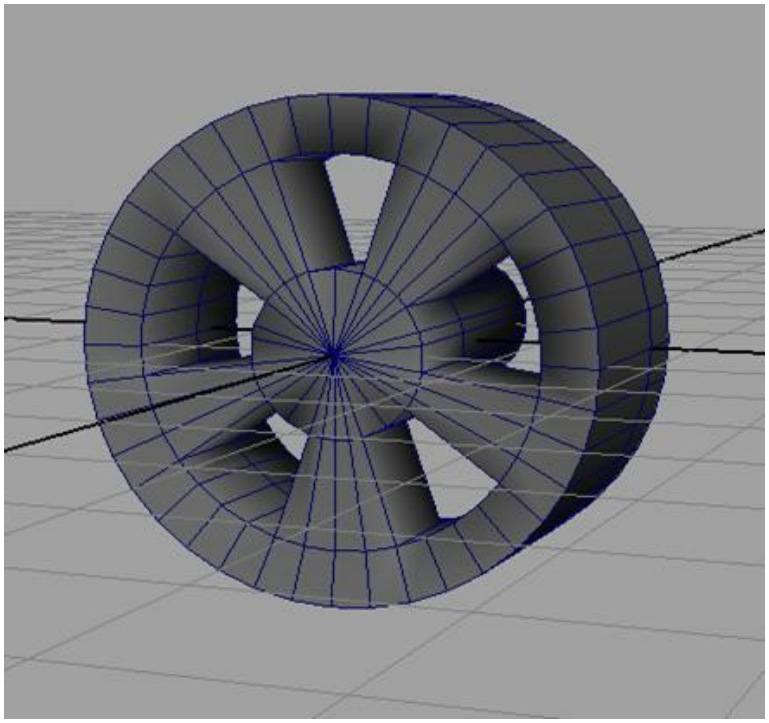


Ilustración 32: Captura de la malla de las ruedas en el modelador

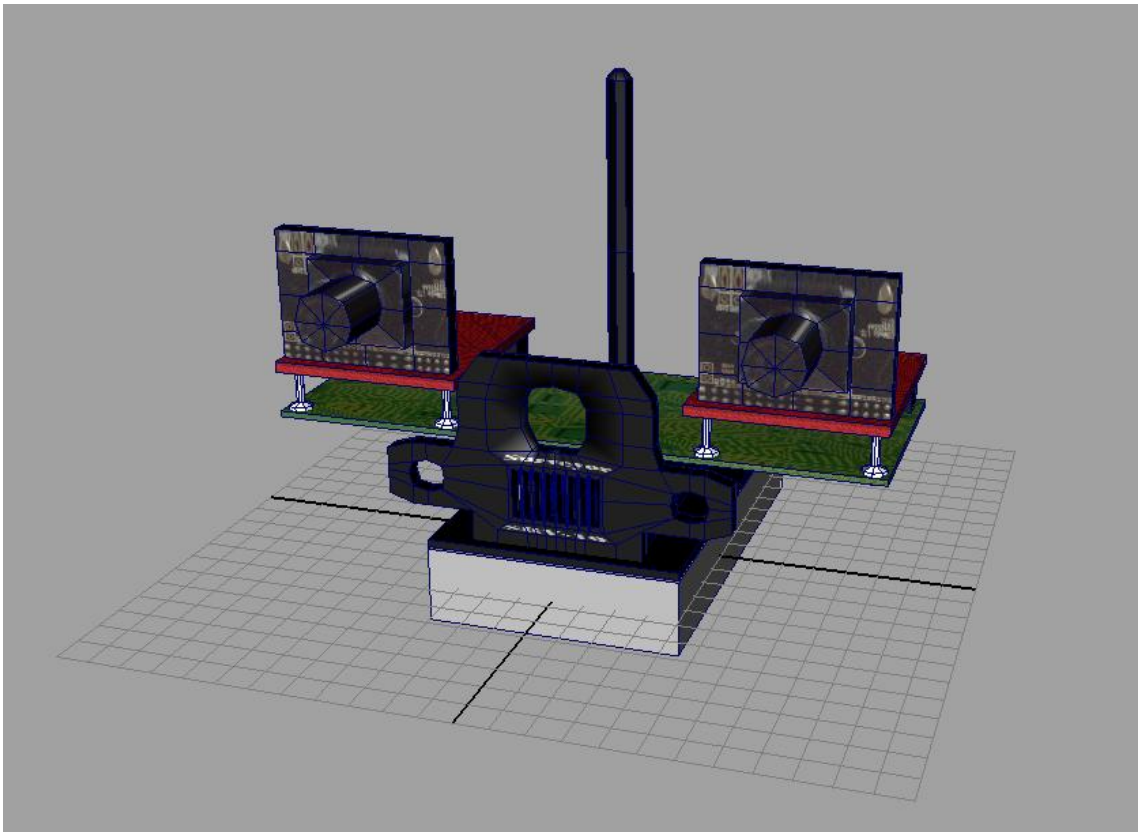


Ilustración 33: Captura de la malla del robot mapeada con textura

8.4.-Entorno de desarrollo

Microsoft Visual Studio es un entorno de desarrollo integrado para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET. Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles.

8.5.-Repositorio

Para el mantenimiento de Proyecto se ha elegido utilizar la herramienta Subversion, es una herramienta de libre distribución que permite el control de versiones de elementos software. Dicho sistema facilita el control de versiones de los

distintos elementos software seleccionados por los miembros del proyecto. Está especialmente orientado a elementos conformados por texto, sobre los que permite realizar un buen manejo, control y gestión de sus modificaciones. Dicha herramienta, en su modo servidor/repositorio, estará alojada en un servicio que Google ofrece, denominado GoogleCode. Se permite, por parte de Google, un tamaño de repositorio de 1 GB (1024 MB).

El repositorio del proyecto en GoogleCode, donde se encuentra la herramienta que hemos desarrollado durante el presente curso académico, es http://code.google.com/p/proyectosistemasinformaticos/

9.-GUIA DE USUARIO

Vamos a ver cómo usar el software.

¿Cómo conectar con el robot?

1. Encienda el robot.
2. Conecte su PC al robot mediante la red wifi, la señal debe ser SRV-1
3. Arranque el software y de al botón Conectar

¿Cómo hacer que funcione el software?

Tiene varias posibilidades según lo que desee hacer:

- Capturar y tratar imágenes del simulador
- Capturar y tratar imágenes del Surveyor
- Crear Obstáculos
- Mover el objetivo por el escenario
- Cambiar de cámara del simulador
- Planificación de caminos
- Búsqueda de bola roja
- Interacción usuario-simulador-Surveyor (Control)

Capturar y tratar imágenes del simulador

Podemos captar las imágenes del simulador desde la perspectiva del robot. Veamos la siguiente captura:

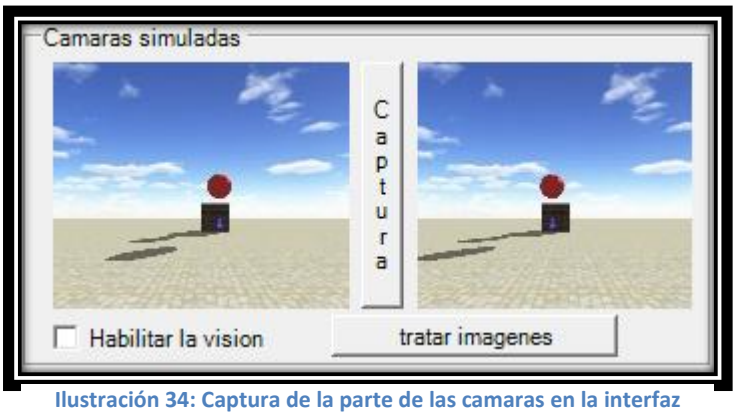


Ilustración 34: Captura de la parte de las camaras en la interfaz

Si damos a captura nos capturará lo que el robot de la simulación ve desde sus dos cámaras. Si damos a tratar, nos reflejará en el log la distancia a la que está nuestro objetivo la posición.

El check Habilitar la visión nos indica que si queremos ver el sentido de visión del robot apareciendo unas líneas desde sus cámaras.

Capturar y tratar imágenes del Surveyor

Podemos captar las imágenes reales del Surveyor, si damos a capturar se guardaran las fotos

Ilustración 35: Captura de la parte de las camaras reales en la interfaz

En esta ocasión tenemos una pelota roja, si damos a capturar y seguidamente a tratar imágenes podremos ver que el simulador sitúa en el entorno 3D el objeto detectado en esta ocasión.

En la siguiente captura vemos el resultado de tratar la imagen anterior, como podemos observar a puesto un objeto a la derecha, justo debajo de donde flota la bola verde. Si comparamos las imágenes reales y las simuladas podemos observar que el objeto se ha colocado a la perfección

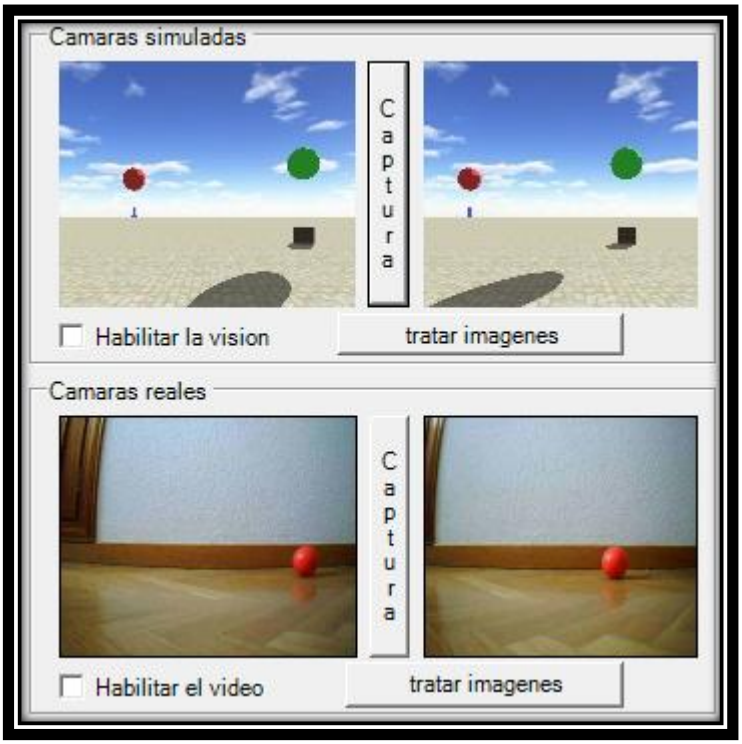


Ilustración 36: Captura de todas las camaras en la interfaz

Crear obstáculos.

En la sección Obstáculos de la interfaz (parte inferior derecha) podemos observar 3 botones

Ilustración 37: Parte de la interfaz dedicada a la modificación del escenario

Las funciones de estos tres botones son las siguientes:

- Crear obstáculos en la simulación
- Limpiar el escenario de la simulación
- Cargar un fichero XML con los obstáculos

Explicaremos más detalladamente cómo crear obstáculos. Si usted pulsa en crear le aparecerá la siguiente pantalla

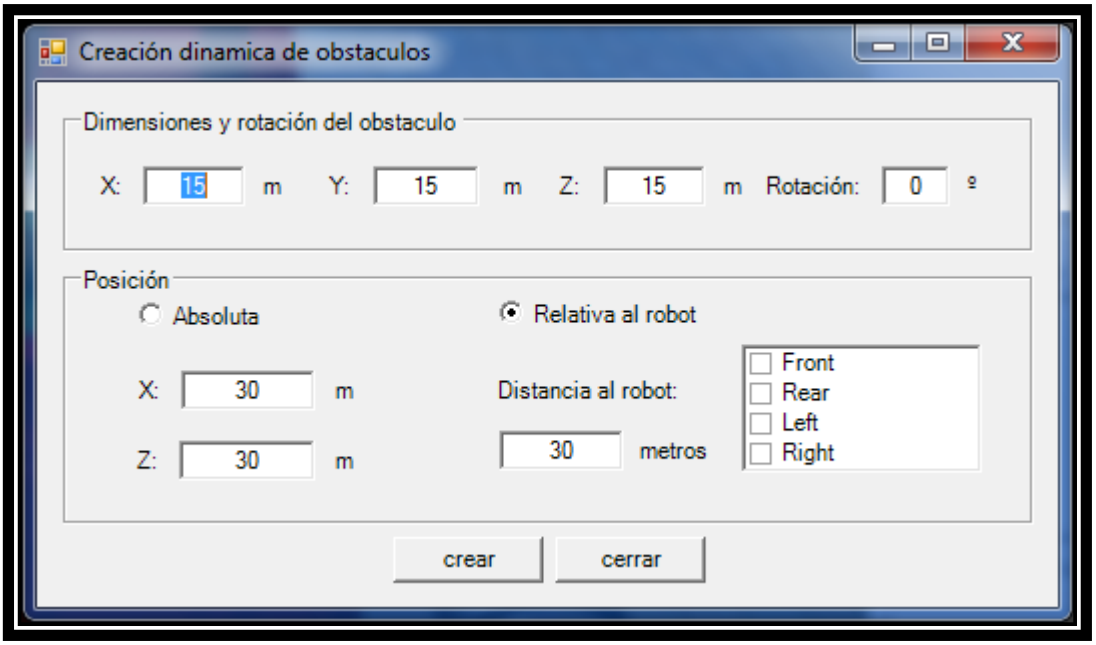


Ilustración 38: Ventana de creación dinamica de obstaculos en el simulador

En ella podremos seleccionar las dimensiones del objeto que se quiere crear y se puede especificar la posición de dos formas distintas, en posiciones (x,z) absolutas del plano o a partir de la posición del robot, se calculará con las opciones introducidas la posición del objeto y se situará.

Mover objetivo

Ilustración 39: Parte de la interfaz dedicada al posicionamiento del destino a alcanzar

Nuestro objetivo o destino será donde se sitúe la flecha, esta flecha podemos situarla en cualquier parte del escenario cómodamente con las flechas de la sección Situación destino

Cambiar de cámara del simulador

Podemos ver diferentes perspectivas lo que se va calculando en la simulación, tenemos 4 tipos diferentes de cámaras.

- Cámara libre
- Cámara 1ªpersona
- Cámara 3ªpersona
- Cámara satélite

Destacamos las cámaras libre y satélite.

En cuanto a la cámara libre hemos de remarcar que con las teclas A, W, S, D podemos mover la cámara y si lo que necesitamos es girar el ángulo de la cámara hemos de hacer un clic sobre la pantalla del simulador y podremos girarla con el ratón, al hacer otro clic fijaremos de nuevo la cámara con el ángulo seleccionado.

En cuanto a la cámara satélite es igual que la cámara libre en cuanto al manejo aunque no se puede girar el ángulo con el ratón.

Planificación de caminos

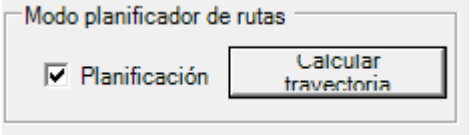


Ilustración 40: Parte de la interfaz para interactuar con el planificador de rutas

Esta sección si pulsamos el botón calcular trayectoria nos calculará la trayectoria viable para llegar al objetivo.

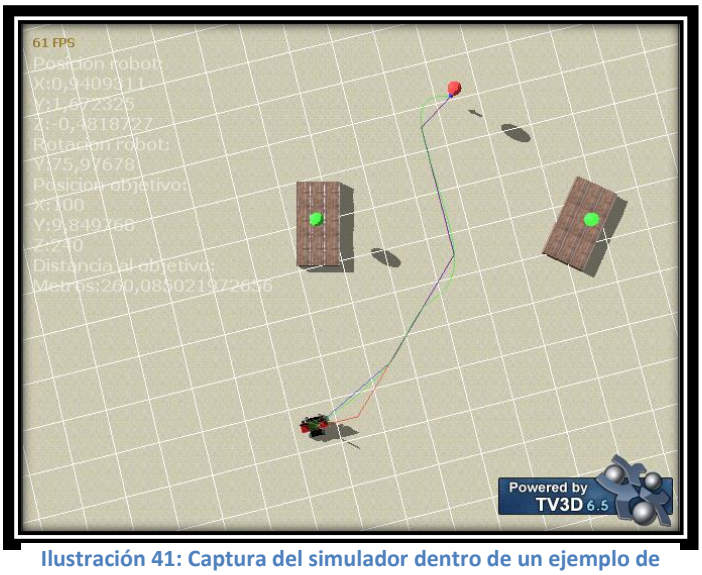


Ilustración 41: Captura del simulador dentro de un ejemplo de simulación

Si seleccionamos Planificación nos aparecerá el grid y la situación del objetivo.

Búsqueda de bola roja

Este es una acción programada como ejemplo a que se podría hacer cualquier otra función, para ello necesitamos tener el Surveyor conectado. Este programa consiste en la búsqueda de la bola roja, el Surveyor dará una vuelta sobre sí mismo en ángulos de 60º buscando la pelota roja, si la localiza programará el camino e irá a su posición.

Interacción usuario-simulador-Surveyor (Control)

Esta sección es una de las más importantes del software a la hora de manejarlo.

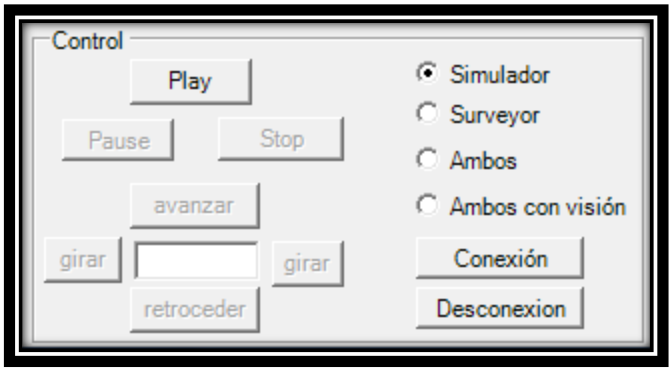


Ilustración 42: Parte de la interfaz para selección de modos e interacción con el robot

Los botones Play, pause y stop son para lanzar, pausar o parar la ejecución de una planificación de caminos.

Los botones avanzar y retroceder son para mandar instrucciones al simulador o al Surveyor o ambos dependiendo de los selectores, en este caso metemos un valor y avanzaremos o retrocederemos ese valor en cm.

Los botones girar sirven para girar a izquierdas o derechas, debemos meter el ángulo a girar, éste lo meteremos en grados.

Como podemos observar tenemos unos selectores, dependiendo de donde estemos haremos una cosa u otra, lo explicamos más detalladamente:

- Simulador: la ejecución de la planificación del camino sólo se realiza con el simulador
- Surveyor: la ejecución de la planificación del camino sólo se realiza con el Surveyor
- Ambos: la ejecución de la planificación del camino se realiza tanto en el simulador como en el Surveyor, en este caso el Surveyor espera al simulador que haga sus movimientos, están sincronizados.
- Ambos con visión: la ejecución de la planificación del camino se realiza tanto en el simulador como en el Surveyor, en este caso el Surveyor espera al simulador que haga sus movimientos, están sincronizados. La diferencia con el anterior reside en que tras cada giro se realiza una captura de imágenes reales, se tratan, se buscan obstáculos y si se han encontrado se vuelve a replanificar la ruta.

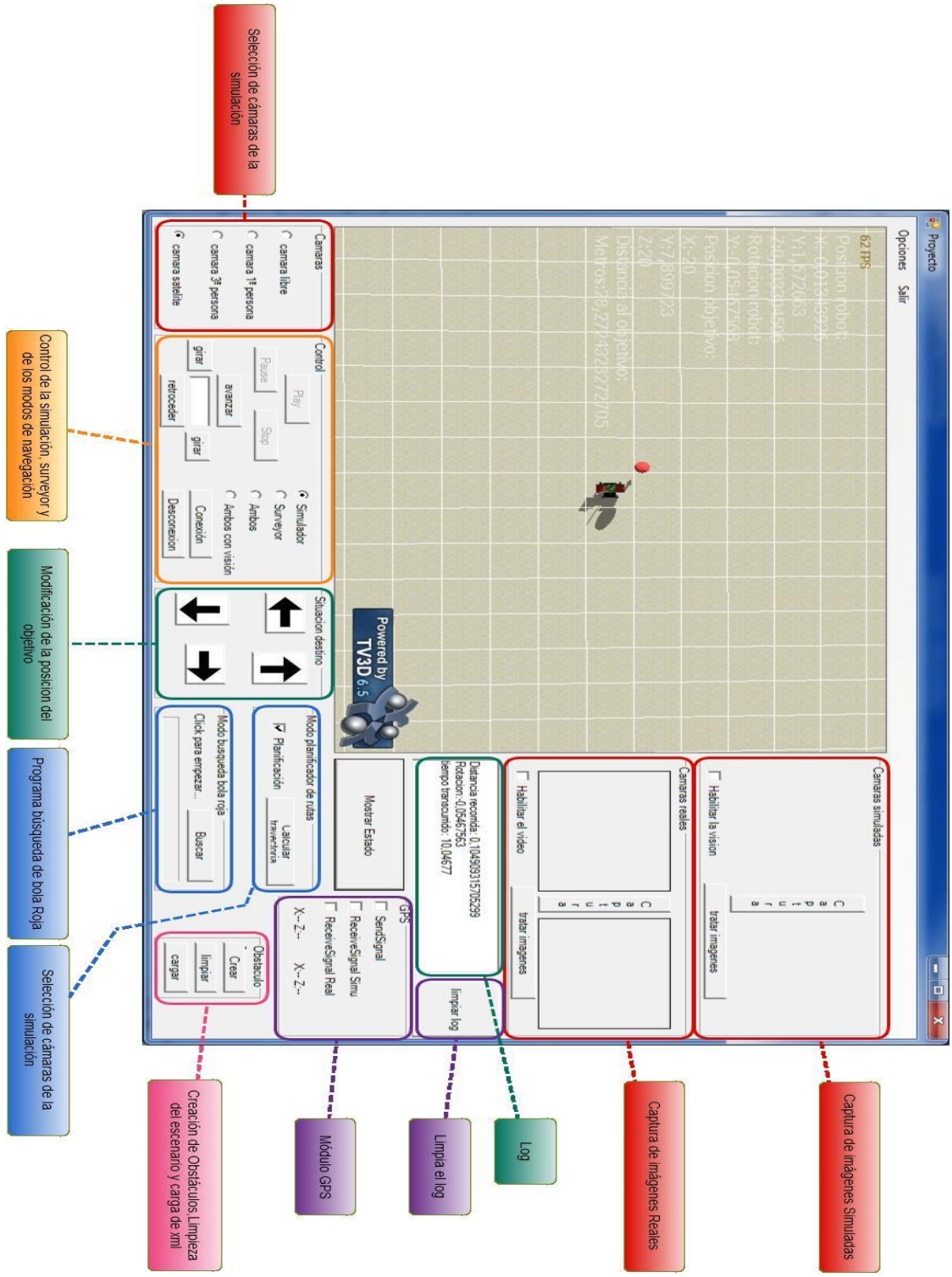
También tenemos en este apartado los botones de conexión y desconexión que sirven para conectar o desconectar el Surveyor al software.

La interacción del usuario con el software para hacer una planificación de un camino con el Surveyor podríamos verla con unos pasos sencillos:

1. Conectar el Surveyor
2. Captar imágenes reales y tratarlas
3. Seleccionar planificación, movemos el objetivo y calculamos ruta.
4. Aquí seleccionaremos un modo, en este caso Ambos con visión.
5. Hacemos clic sobre Play.

Podremos ver cómo el software planifica y manda las instrucciones necesarias al robot para llegar a su destino sin chocar con el obstáculo.

Vista general de la interfaz



10.-RESULTADOS

Se han realizado multitud de pruebas para probar el correcto funcionamiento de todos los módulos integrados en la aplicación, a continuación mostramos unas cuantas pruebas relacionadas con las principales funcionalidades del sistema y la información recopilada por cada una de ellas.

Partimos desde la prueba más sencilla con la cual se cubría el primer objetivo hasta las distintas funcionalidades que se han ido añadiendo a continuación.

10.1.-Prueba 1: Ambos con visión

Objetivo: Esquivar objeto.

Modo de ejecución: Ambos

Dificultad: Posicionamiento del objeto en el entorno 3D por visión estereoscópica. Planificación de la ruta.

En esta primera prueba demostramos el primer objetivo que se estableció en el proyecto, evitar la colisión de un objeto. Hay que destacar que esta prueba ya se pone un 80% del código en funcionamiento, puesto que se usa el simulador, la conexión via WiFi con el SRV-1,la visión estereoscópica con el cálculo de la posición del objetivo relativa al robot, la planificación del camino en la simulación y en el SRV-1, y la ejecución de esta planificación de manera correcta.

Resultados de la prueba nº 1	
Detalles de ejecución	
Modo de ejecución:	Ambos con visión
GPS activado:	No
Algoritmo de detección de obstáculos:	Algoritmo basado en colores
Alcanza el objetivo:	Sí
Descripción textual	
Colocamos enfrente del robot un único obstáculo, en concreto, una bola roja. Ponemos la posición objetivo de la ruta detrás del obstáculo.	
Datos de la ruta	
Posición origen:	(X: 0, Z: 0)
Rotación inicial:	179,9951

Posición objetivo:			(X: -170, Z: 0)
Distancia al objetivo (distancia simulador):			170
Lista de pasos original			
Paso 1			
Nº de acción	**Tipo**	**Cantidad**	**Observaciones**
1	Girar	0,00428772	
2	Consultar giro	-	GPS no activado => No consulta
3	Visión	-	Nuevo obstáculo => Replanifica
4	Avanzar	169,9583	Replanificación => No se ejecuta
5	Consultar	-	Replanificación => No se ejecuta
6	Sincronismo	-	Replanificación => No se ejecuta
Lista de pasos tras la replanificación			
Paso 1			
Nº de acción	**Tipo**	**Cantidad**	**Observaciones**
1	Girar	45	
2	Consultar giro	-	GPS no activado => No consulta
3	Visión	-	No replanifica
4	Avanzar	90,50968	
5	Consultar	-	GPS no activado => No consulta
Paso 2			
Nº de acción	**Tipo**	**Cantidad**	**Observaciones**
1	Girar	-45	
2	Consultar giro	-	GPS no activado => No consulta
3	Visión	-	No replanifica
4	Avanzar	64	
5	Consultar	-	GPS no activado => No consulta
Paso 3			
Nº de acción	**Tipo**	**Cantidad**	**Observaciones**
1	Girar	-45,00002	
2	Consultar giro	-	GPS no activado => No consulta
3	Visión	-	No replanifica
4	Avanzar	45,25483	
5	Consultar	-	GPS no activado => No consulta
Paso 4			
Nº de acción	**Tipo**	**Cantidad**	**Observaciones**

1	Girar	-27,7093	
2	Consultar giro	-	GPS no activado => No consulta
3	Visión	-	No replanifica
4	Avanzar	33,50158	
5	Consultar	-	GPS no activado => No consulta
6	Sincronismo	-	

Ilustración 43: Prueba 1. Situación inicial.

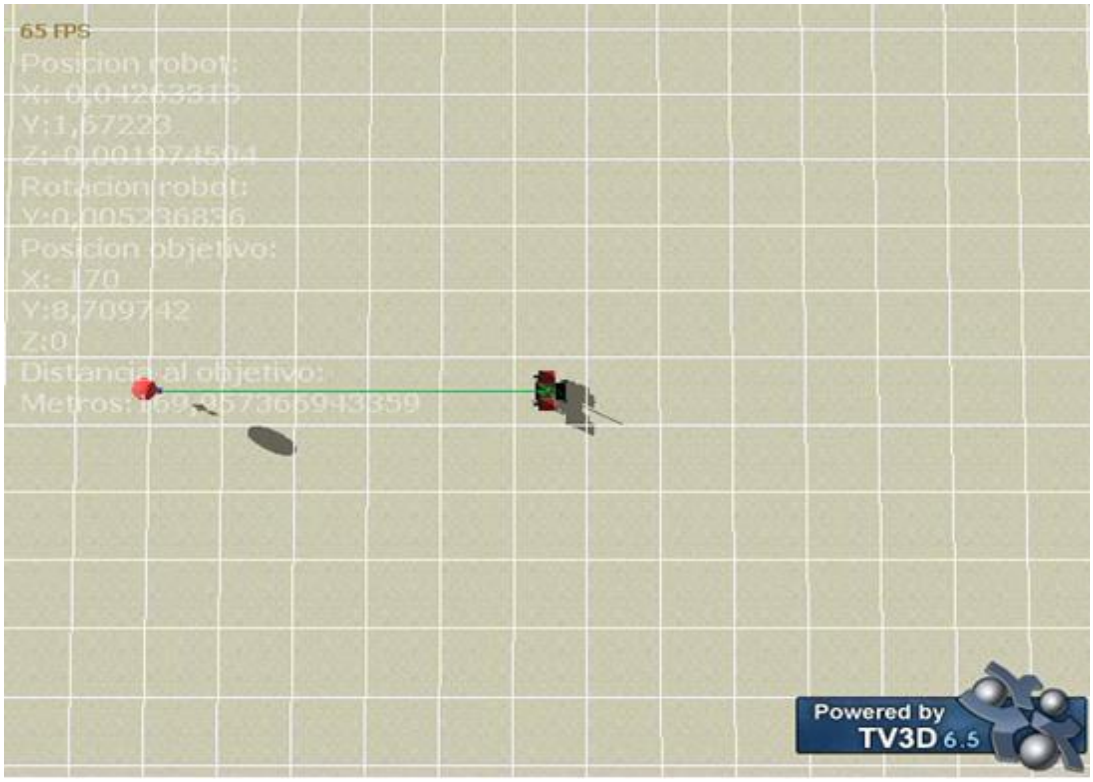


Ilustración 44: Prueba 1. Situación en el simulador al capturar las instantáneas de la acción de visión 1, inmediatamente antes de la replanificación. Observación: En la simulación la bola roja es el objetivo, el obstáculo todavía no lo ha encontrado y, por ello, la primera ruta calculada es una línea recta.

Ilustración 45: Prueba 1. Instantánea de la cámara izquierda en la acción de visión 1. Aparece el obstáculo.

Ilustración 46: Prueba 1. Instantánea de la cámara derecha en la acción de visión 1. Aparece el obstáculo.

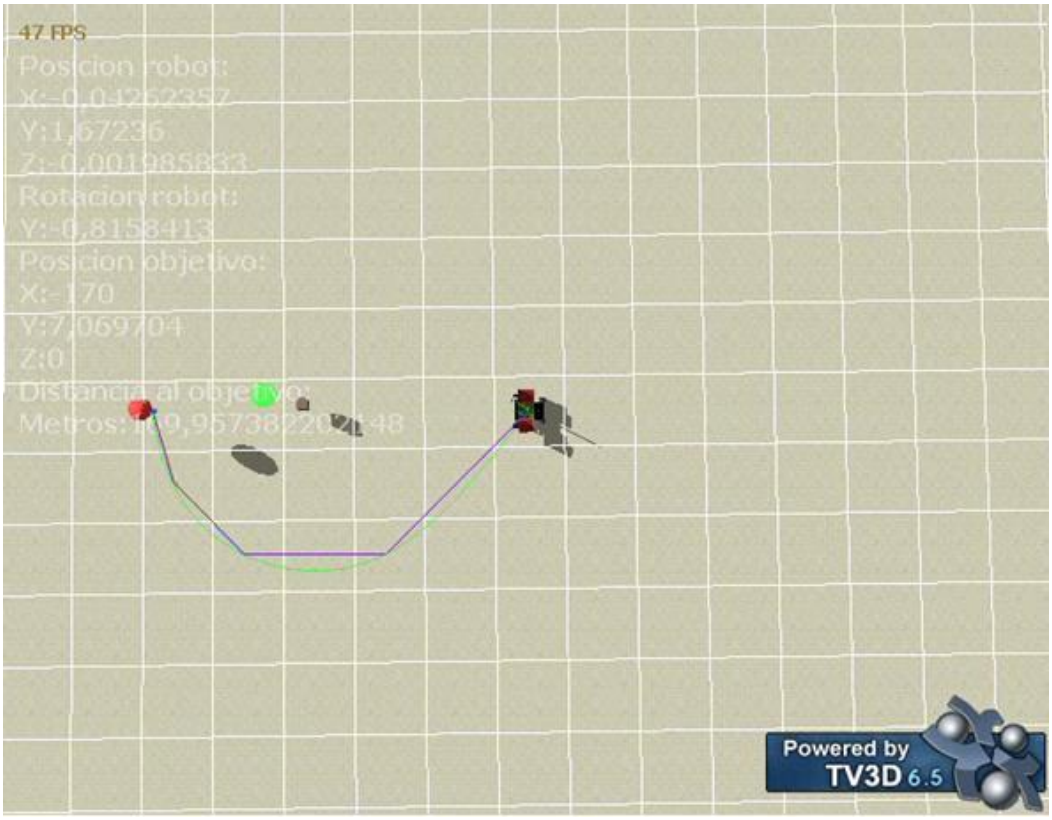


Ilustración 47: Prueba 1. En la acción de visión 1 aparece el obstáculo, al tratarse las imágenes se comprueba que se trata de un obstáculo nuevo, lo que hace que se realice una replanificación.

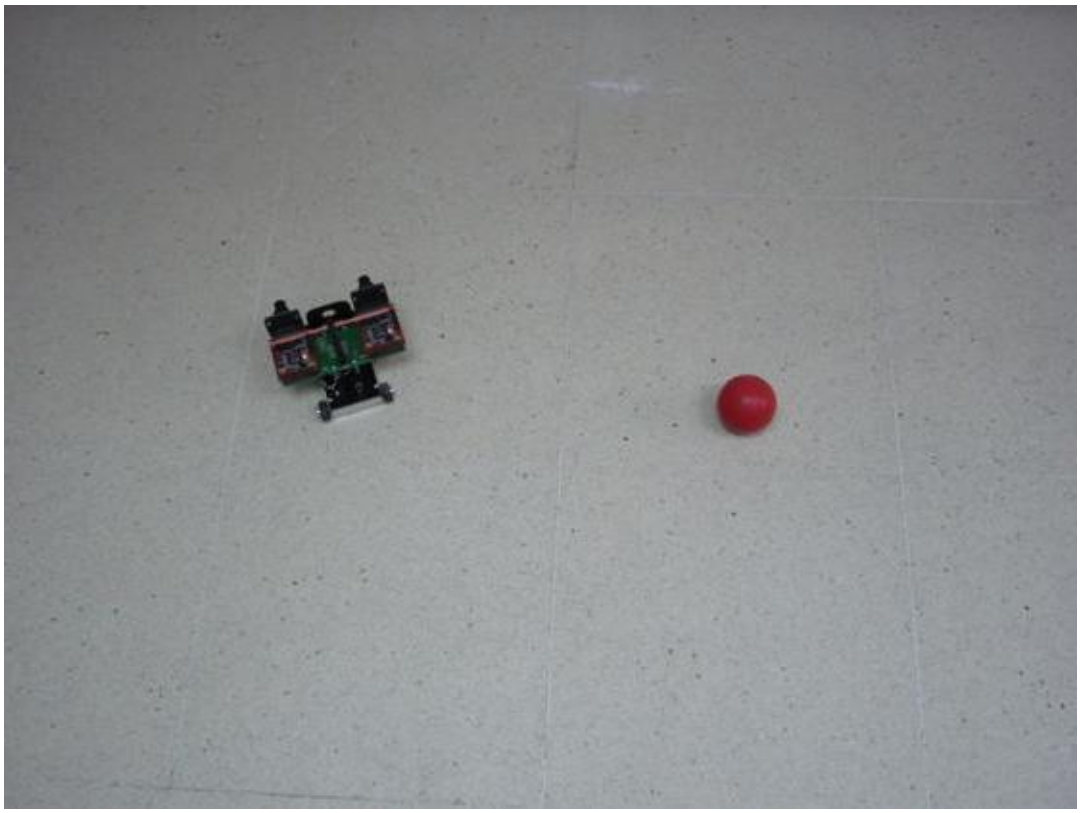


Ilustración 48: Prueba 1. Situación real al final de la ruta

10.2.-Prueba 2: Búsqueda de bola roja

Objetivo: Encontrar una bola roja. Damos pequeños giros de 60º hasta que localizamos el objetivo o damos una vuelta completa. Este objetivo demuestra el control de rotación estimada de navegación sin GPS.

Modo de ejecución: Ambos

Dificultad: Control de rotación del surveyor para situar correctamente el objetivo en el entorno de la simulación para efectuar correctamente la planificación del camino.

En esta prueba demostramos que se puede programar cualquier tipo de ejecución predefinida y se demuestra que pese a no tener GPS puede situar el objetivo de manera bastante precisa. En este caso tenemos una bola roja que localizar e ir hasta ella.

Resultados de la prueba nº 2			
Detalles de ejecución			
Modo de ejecución:	Bola roja		
GPS activado:	No		
Algoritmo de detección de obstáculos:	Algoritmo basado en colores		
Encuentra la bola roja:	Sí		
Descripción textual			
El robot debe buscar la bola roja en un escenario sin obstáculos. Cuando la encuentra, se acerca a ella.			
Datos de la ruta			
Posición inicial:	(X: 0, Z: 0)		
Rotación inicial:	179,9951		
Búsqueda			
Nº de capturas	**Giro acumulado**	**Encuentra bola roja**	**¿Obstáculo?**
1	0º	No	No
2	60º	No	No
3	120º	No	No
4	180º	No	No
5	240º	Sí => Planifica acercamiento	No
Acercamiento. Lista de pasos tras encontrar la bola roja			
Paso 1			

Nº de acción	Tipo	Cantidad	Observaciones
1	Girar	-18,9098	
2	Consultar giro	-	GPS no activado => No consulta
3	Avanzar	153,0396	
4	Consultar	-	GPS no activado => No consulta
5	Sincronismo	-	

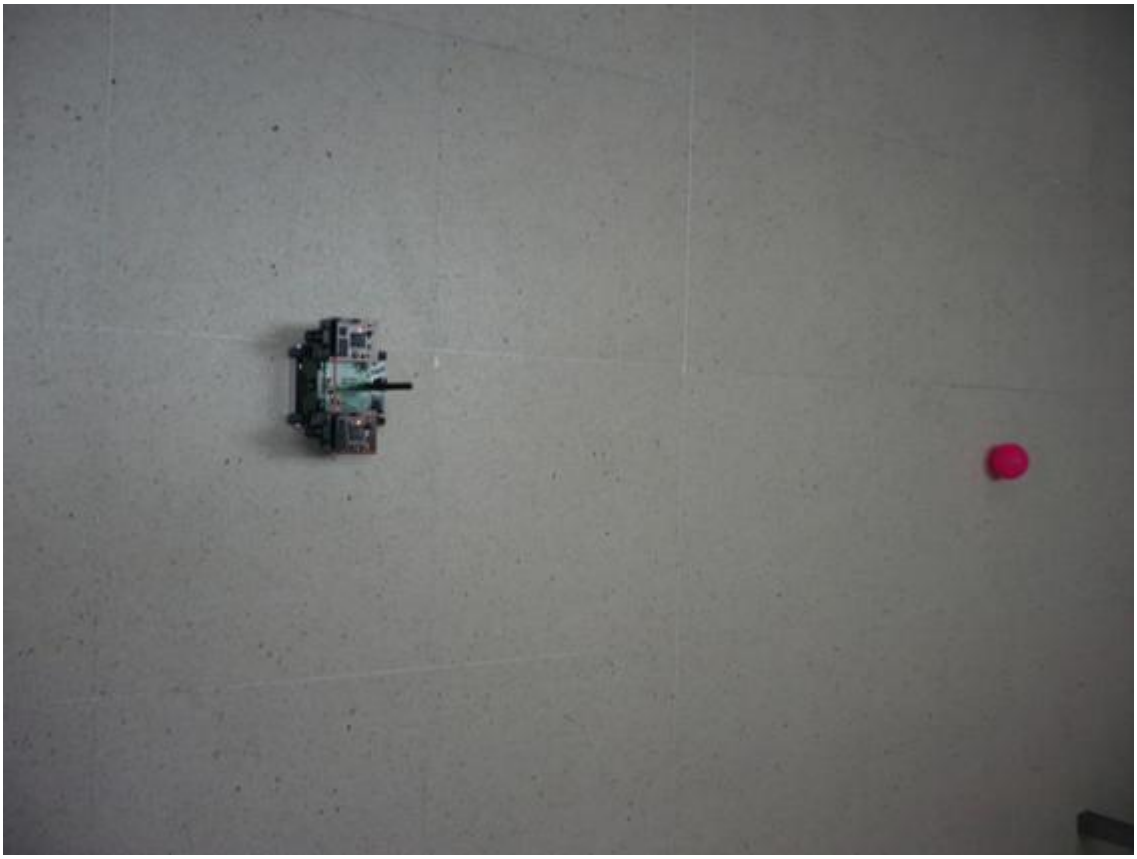


Ilustración 49: Prueba 2. Situación inicial.

Ilustración 50: Prueba 2. Instantánea de la cámara izquierda en la captura nº 5 de la búsqueda.

Ilustración 51: Prueba 2. Instantánea de la cámara derecha en la captura nº 5 de la búsqueda.

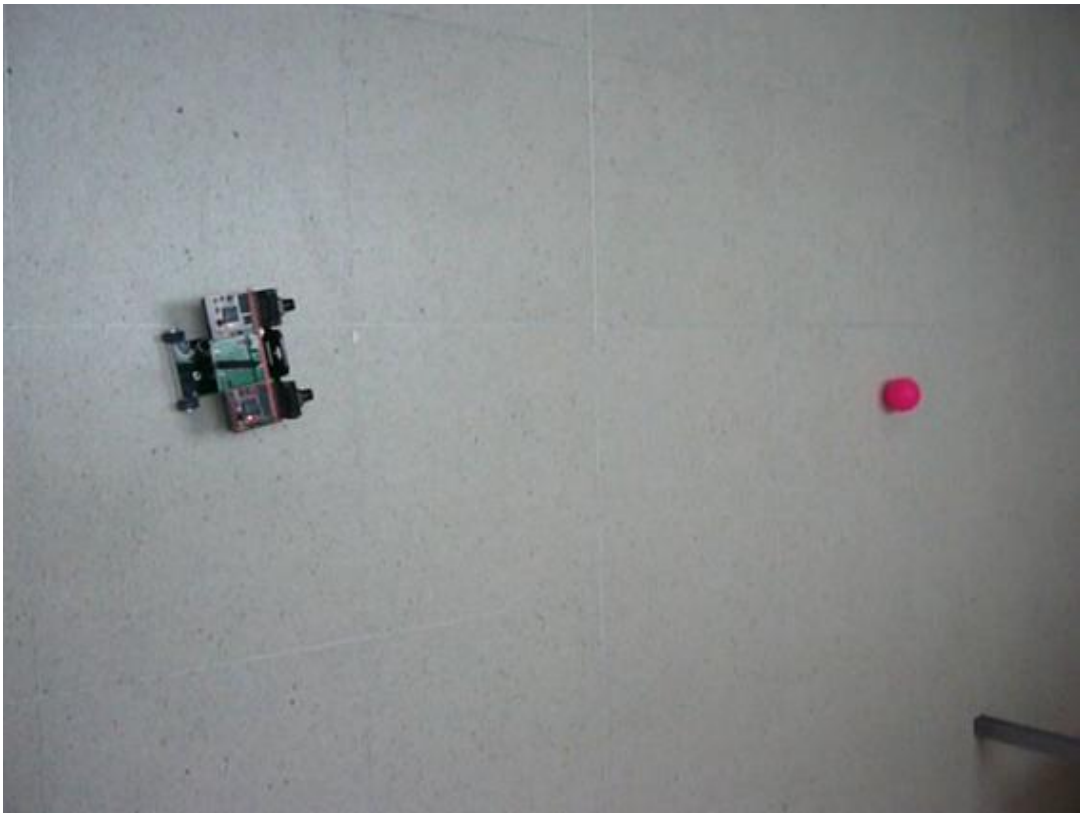


Ilustración 52: Prueba 2. Situación del robot real al encontrar la bola roja

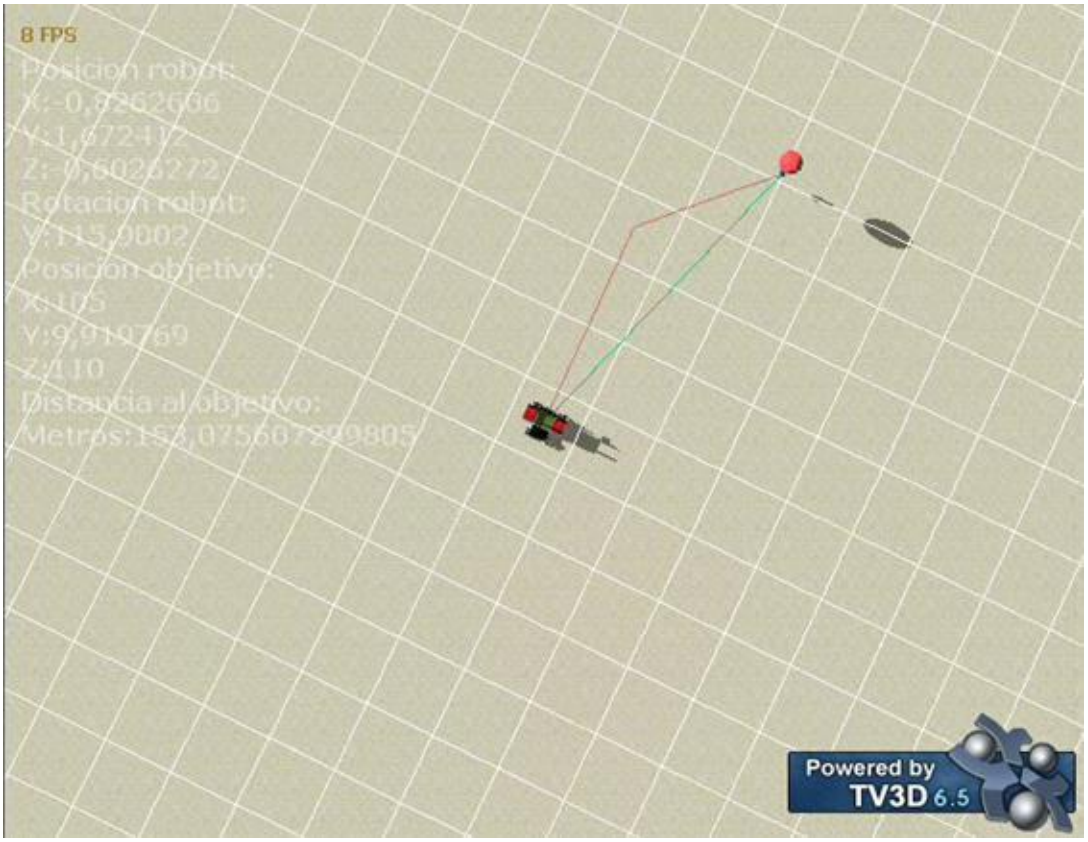


Ilustración 53: Prueba 2. Planificación para alcanzar la bola roja

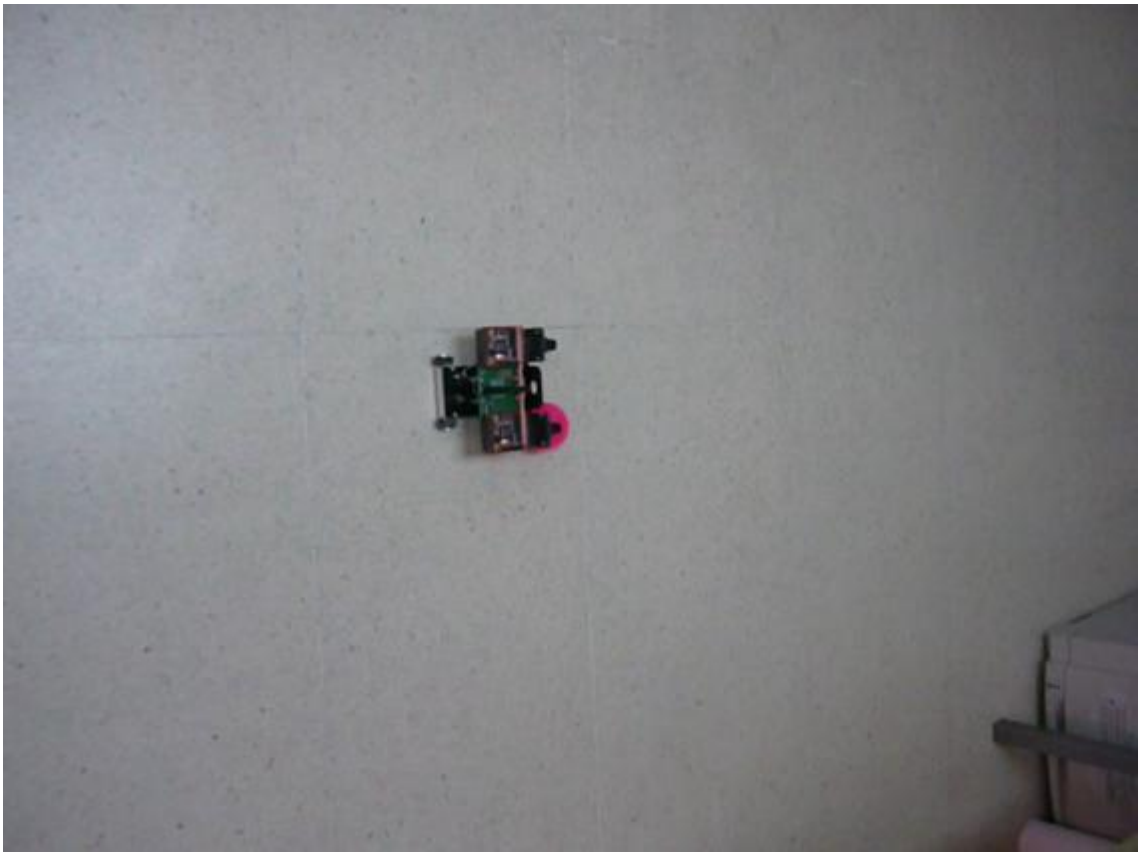


Ilustración 54: Prueba 2. Situación final.

10.3.-Prueba 3: Búsqueda de bola roja con obstáculos

Objetivo: Encontrar una bola roja. Damos pequeños giros de 60º hasta que localizamos el objetivo o damos una vuelta completa. Este objetivo demuestra el control de rotación estimada de navegación sin GPS y la precisión de la situación de obstáculos en el entorno 3D.

Modo de ejecución: Ambos

Dificultad: Control de rotación del surveyor para situar correctamente el objetivo en el entorno de la simulación para efectuar correctamente la planificación del camino.

En este caso tenemos una prueba parecida a la anterior pero con un nivel más de dificultad: hay obstáculos entre medias que hay que evitar si no queremos colisionar para ello después de analizar cada foto y saber si hay obstáculos o saber si está nuestro objetivo (bola roja) planificamos el camino correcto.

Resultados de la prueba nº 3			
Detalles de ejecución			
Modo de ejecución:	Bola roja		
GPS activado:	No		
Algoritmo de detección de obstáculos:	Algoritmo basado en colores		
Encuentra la bola roja:	Sí		
Descripción textual			
El robot debe buscar y acercarse a la bola roja en un escenario en el que hay un obstáculo.			
Datos de la ruta			
Posición inicial:	(X: 0, Z: 0)		
Rotación inicial:	179,9951		
Búsqueda			
Nº de capturas	**Giro acumulado**	**Encuentra bola roja**	**¿Obstáculo?**
1	0º	No	No
2	60º	No	No
3	120º	No	No
4	180º	Sí => Planifica acercamiento	Sí
Acercamiento. Lista de pasos tras encontrar la bola roja			
Paso 1			

Nº de acción	Tipo	Cantidad	Observaciones
1	Girar	-44,99998	
2	Consultar giro	-	GPS no activado => No consulta
3	Avanzar	45,2583	
4	Consultar	-	GPS no activado => No consulta
Paso 2			
Nº de acción	**Tipo**	**Cantidad**	**Observaciones**
1	Girar	-0,000003	
2	Consultar giro	-	GPS no activado => No consulta
3	Avanzar	45,25483	
4	Consultar	-	GPS no activado => No consulta
Paso 3			
Nº de acción	**Tipo**	**Cantidad**	**Observaciones**
1	Girar	45	
2	Consultar giro	-	GPS no activado => No consulta
3	Avanzar	64	
4	Consultar	-	GPS no activado => No consulta
Paso 4			
Nº de acción	**Tipo**	**Cantidad**	**Observaciones**
1	Girar	40,23401	
2	Consultar giro	-	GPS no activado => No consulta
3	Avanzar	35,04219	
4	Consultar	-	GPS no activado => No consulta
5	Sincronismo	-	

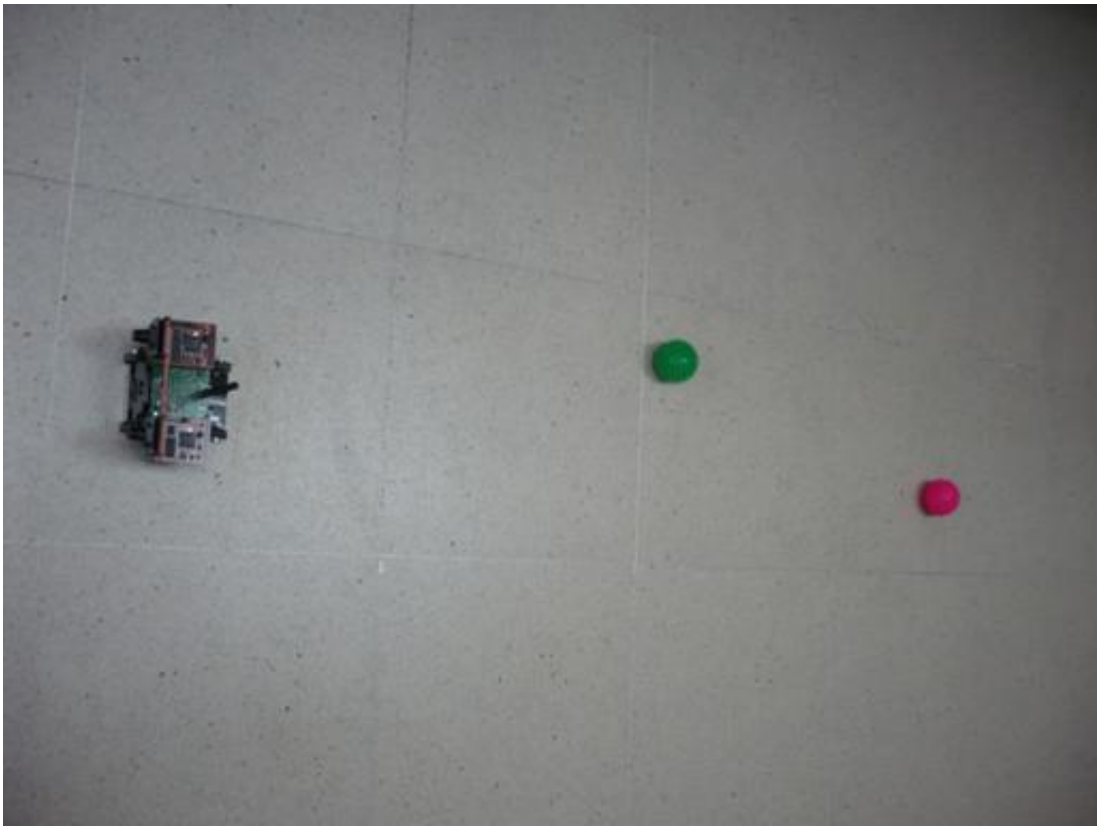


Ilustración 55: Prueba 3. Situación inicial.

Ilustración 56: Prueba 3. Instantánea de la cámara izquierda en la captura nº 4 de la búsqueda.

Ilustración 57: Prueba 3. Instantánea de la cámara derecha en la captura nº 4 de la búsqueda.

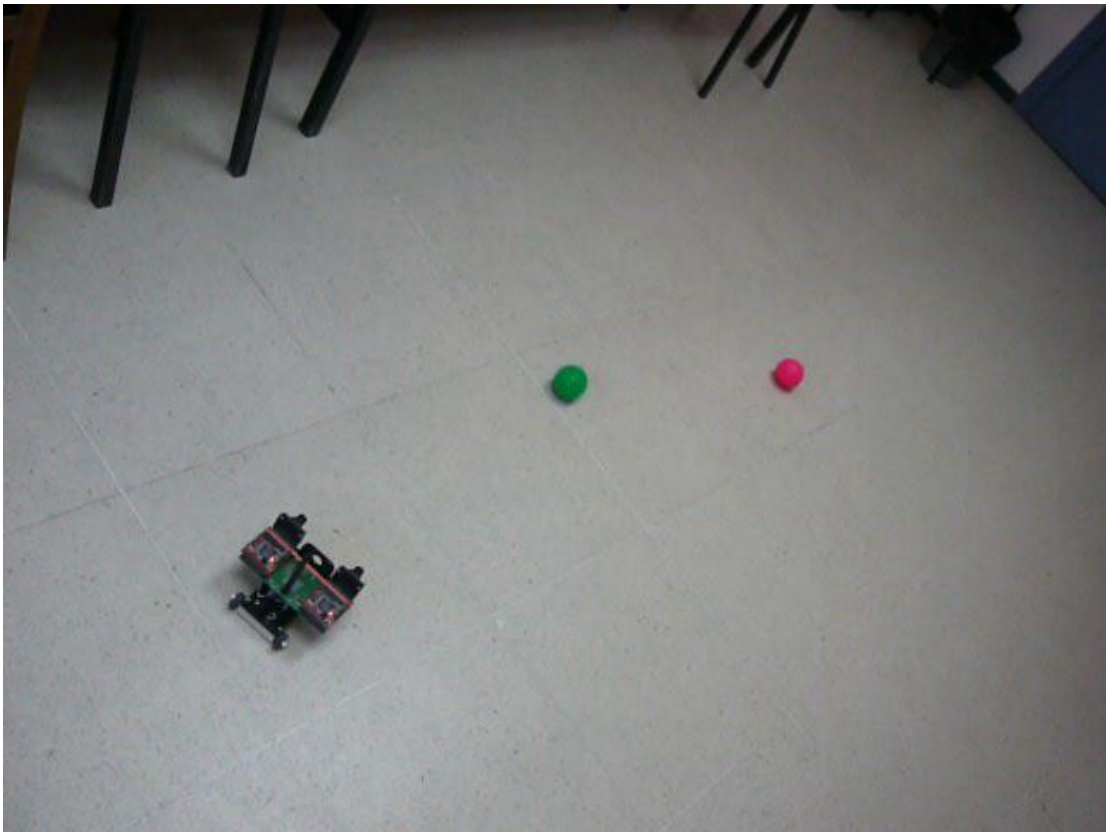


Ilustración 58: Prueba 3. Situación del robot real al encontrar la bola roja

Ilustración 59: Prueba 3. Planificación para alcanzar la bola roja

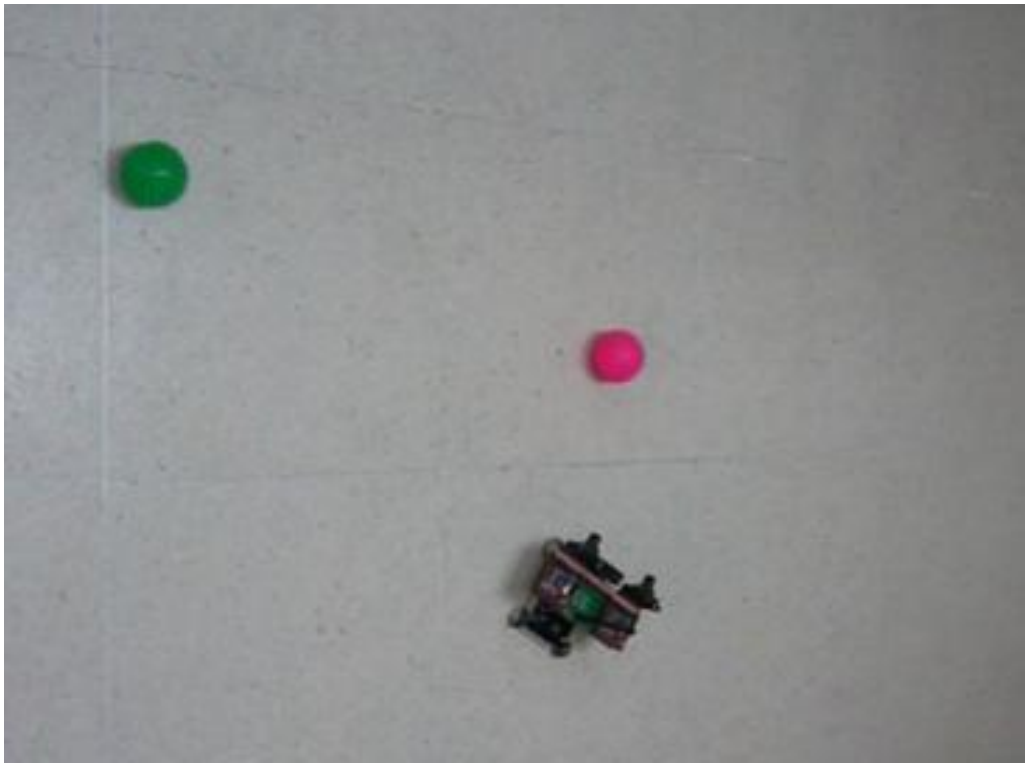


Ilustración 60: Prueba 3. Situación final.

Ilustración 61: Prueba 3. Situación final en el simulador

10.4.-Prueba 4: Trayectoria con obstáculos por descubrir.

Objetivo: Llegar a la coordenada indicada detectando obstáculos por el camino a través de la visión estereoscópica.

Modo de ejecución: Ambos con visión

Dificultad: Análisis de visión estereoscópica tras cada giro, detección de obstáculos ya analizados antes, detección de obstáculos nuevos y re-planificación del camino en caso que sea necesario. Otra dificultad a destacar es el sincronismo entre simulación y SRV-1.

Demostración de la evolución al objetivo principal del proyecto. Visión estereoscópica en funcionamiento tras cada giro, lo que permite que podamos trazar cualquier camino de una coordenada a otra sin problema de colisión debido a la detección de obstáculos nuevos en cada giro. Trazaremos un camino, tras cada giro haremos una foto para poder analizar nuevos obstáculos no detectados anteriormente, en el caso de que los haya, re-planificaremos el camino de tal forma que evitemos esos obstáculos hasta llegar a nuestro objetivo.

Resultados de la prueba nº 4	
Detalles de ejecución	
Modo de ejecución:	Ambos con visión
GPS activado:	No
Algoritmo de detección de obstáculos:	Algoritmo basado en colores
Alcanza el objetivo:	Sí
Descripción textual	
Colocamos enfrente del robot dos obstáculos: una bola verde y una roja. La bola verde está encima de una caja que oculta la bola roja al principio. Así el robot, verá primero la bola verde y más tarde la bola roja. Esto hará que replanifique en dos ocasiones. Ponemos la posición objetivo de la ruta detrás del último obstáculo. La posición objetivo la marcamos con una bola azul.	
Datos de la ruta	
Posición origen:	(X: 0, Z: 0)
Rotación inicial:	179,9853
Posición objetivo:	(X: -420, Z: -20)
Distancia al objetivo (distancia simulador):	420,42
Lista de pasos original	
Paso 1	

Nº de acción	Tipo	Cantidad	Observaciones
1	Girar	2,742004	
2	Consultar giro	-	GPS no activado => No consulta
3	Visión	-	Nuevo obstáculo => Replanifica
4	Avanzar	420,4497	Replanificación => No se ejecuta
5	Consultar	-	Replanificación => No se ejecuta
6	Sincronismo	-	Replanificación => No se ejecuta
Lista de pasos tras la primera replanificación			
Paso 1			
Nº de acción	**Tipo**	**Cantidad**	**Observaciones**
1	Girar	33,69008	
2	Consultar giro	-	GPS no activado => No consulta
3	Visión	-	No replanifica
4	Avanzar	115,3777	
5	Consultar	-	GPS no activado => No consulta
Paso 2			
Nº de acción	**Tipo**	**Cantidad**	**Observaciones**
1	Girar	-33,69006	
2	Consultar giro	-	GPS no activado => No consulta
3	Visión	-	Nuevo obstáculo => Replanifica
4	Avanzar	256,0001	Replanificación => No se ejecuta
5	Consultar	-	Replanificación => No se ejecuta
Paso 3			
Nº de acción	**Tipo**	**Cantidad**	**Observaciones**
1	Girar	-44,66165	Replanificación => No se ejecuta
2	Consultar giro	-	Replanificación => No se ejecuta
3	Visión	-	Replanificación => No se ejecuta
4	Avanzar	96,18596	Replanificación => No se ejecuta
5	Consultar	-	Replanificación => No se ejecuta
6	Sincronismo	-	Replanificación => No se ejecuta

Lista de pasos tras la segunda replanificación			
Paso 1			
Nº de acción	**Tipo**	**Cantidad**	**Observaciones**
1	Girar	0	
2	Consultar giro	-	GPS no activado => No consulta
3	Visión	-	No replanifica
4	Avanzar	96,00005	
5	Consultar	-	GPS no activado => No consulta
Paso 2			
Nº de acción	**Tipo**	**Cantidad**	**Observaciones**
1	Girar	-45	
2	Consultar giro	-	GPS no activado => No consulta
3	Visión	-	No replanifica
4	Avanzar	45,25484	
5	Consultar	-	GPS no activado => No consulta
Paso 3			
Nº de acción	**Tipo**	**Cantidad**	**Observaciones**
1	Girar	0	
2	Consultar giro	-	GPS no activado => No consulta
3	Visión	-	No replanifica
4	Avanzar	45,25484	
5	Consultar	-	GPS no activado => No consulta
Paso 4			
Nº de acción	**Tipo**	**Cantidad**	**Observaciones**
1	Girar	0	
2	Consultar giro	-	GPS no activado => No consulta
3	Visión	-	No replanifica
4	Avanzar	45,25486	
5	Consultar	-	GPS no activado => No consulta
Paso 5			
Nº de acción	**Tipo**	**Cantidad**	**Observaciones**
1	Girar	45	
2	Consultar giro	-	GPS no activado => No consulta
3	Visión	-	No replanifica
4	Avanzar	96,00002	
5	Consultar	-	GPS no activado => No consulta
Paso 6			

Nº de acción	Tipo	Cantidad	Observaciones
1	Girar	45,27043	
2	Consultar giro	-	GPS no activado => No consulta
3	Visión	-	No replanifica
4	Avanzar	51,45098	
5	Consultar	-	GPS no activado => No consulta
6	Sincronismo	-	

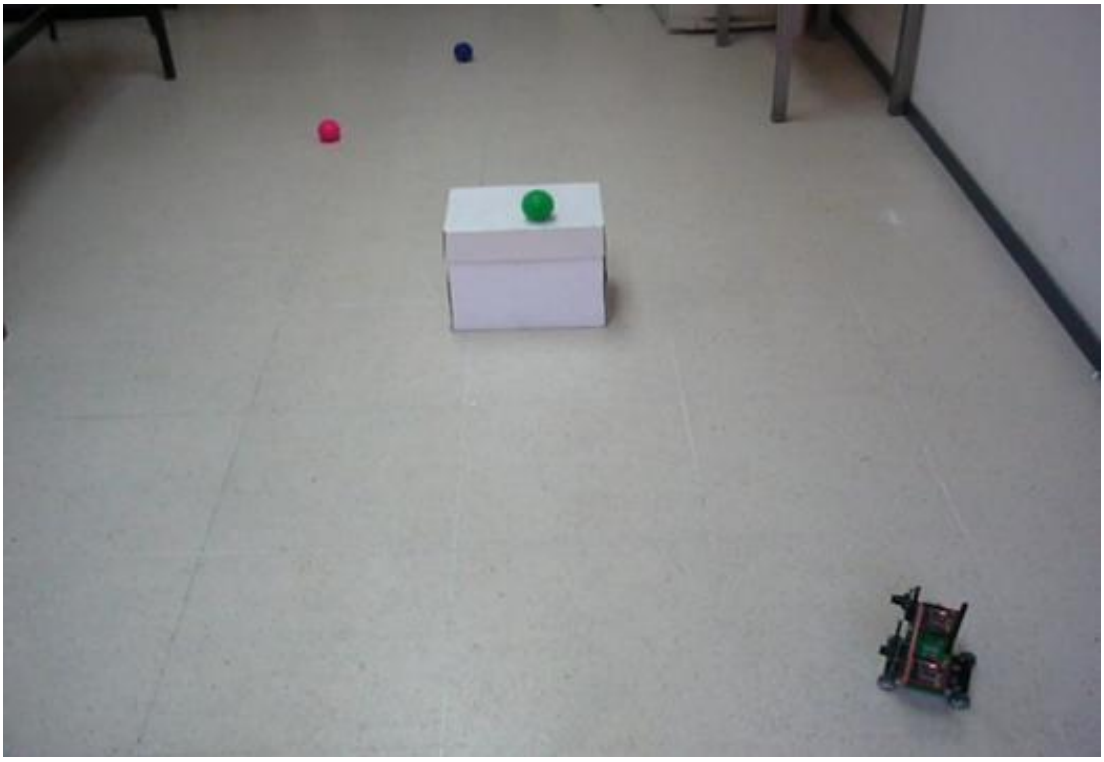


Ilustración 62: Prueba 4. Situación inicial.

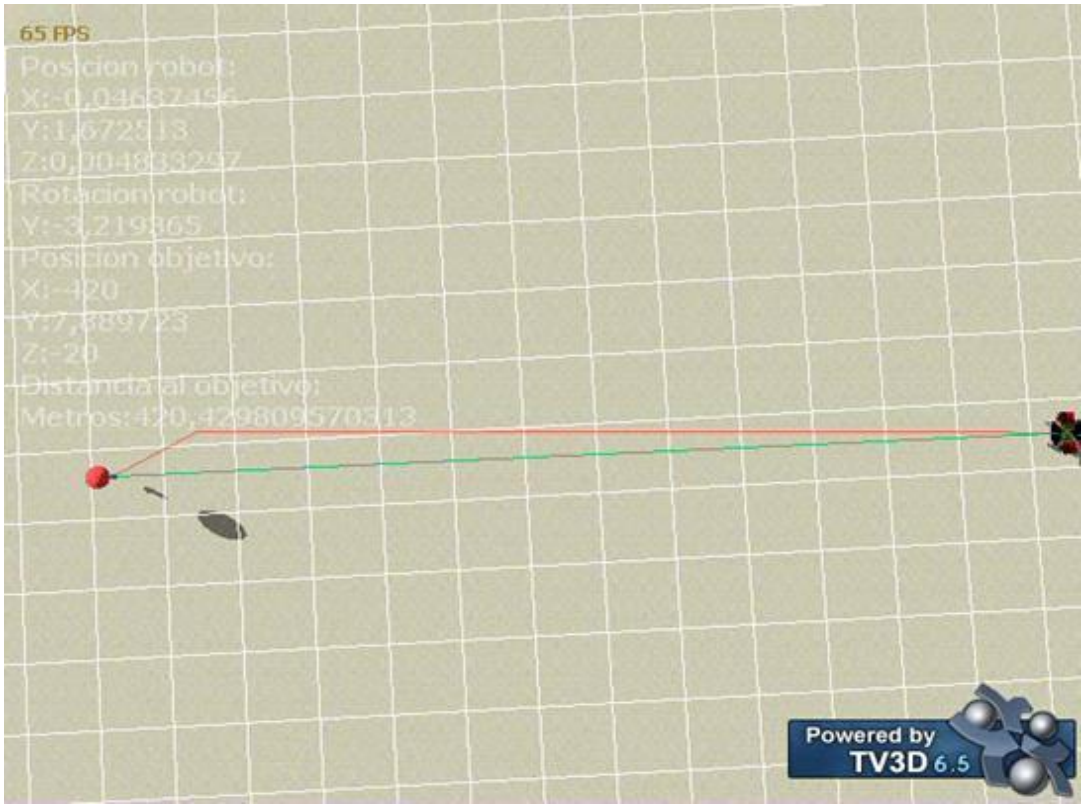


Ilustración 63: Prueba 4. Situación en el simulador al capturar las instantáneas de la acción de visión 1, inmediatamente antes de la replanificación. Observación: En la simulación la bola roja es el objetivo, ningún obstáculo ha sido encontrado por el momento y, por ello, la primera ruta calculada es una línea recta.

Ilustración 64: Prueba 4. Instantánea de la cámara izquierda en la acción de visión 1. Aparece el primer obstáculo.

Ilustración 65: Prueba 4. Instantánea de la cámara derecha en la acción de visión 1. Aparece el primer obstáculo.

Ilustración 66: Prueba 4. En la acción de visión 1 aparece un obstáculo nuevo y replanifica.

Ilustración 67: Prueba 4. Instantánea de la cámara izquierda en la acción de visión 3. Aparece un obstáculo nuevo.

Ilustración 68: Prueba 4. Instantánea de la cámara derecha en la acción de visión 3. Aparece un obstáculo nuevo.

Ilustración 69: Prueba 4. En la acción de visión 3 aparece el otro obstáculo y tras tratar las imágenes, replanifica.

Ilustración 70: Prueba 4. Instantánea de la cámara izquierda en la acción de visión 4. El obstáculo que aparece ya ha sido detectado previamente y, por ello, no se replanifica.

Ilustración 71: Prueba 4. Instantánea de la cámara derecha en la acción de visión 4. El obstáculo que aparece ya ha sido detectado previamente y, por ello, no se replanifica.

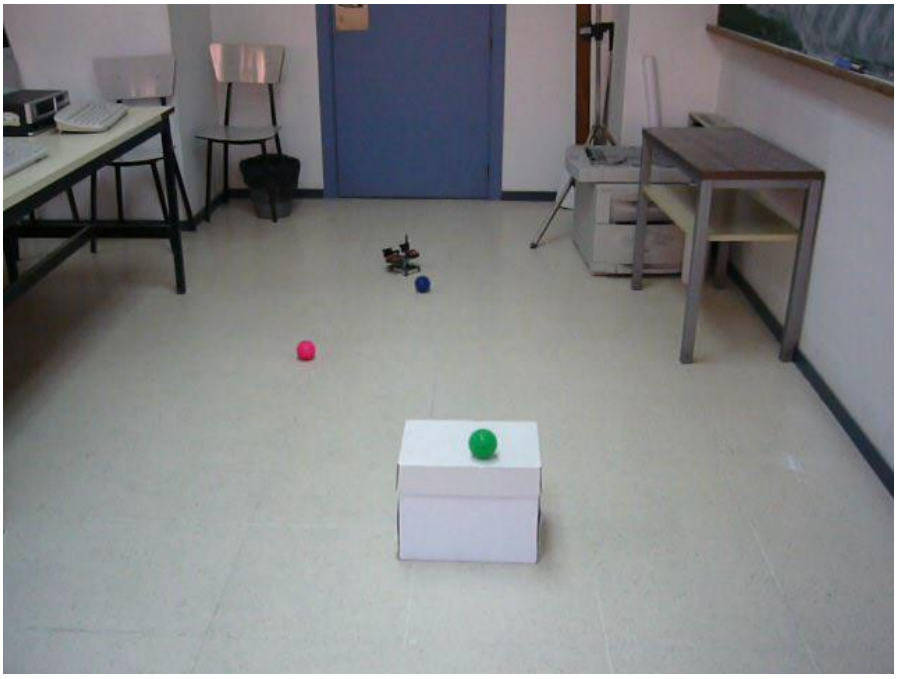


Ilustración 72: Prueba 4. Situación real al final de la ruta.

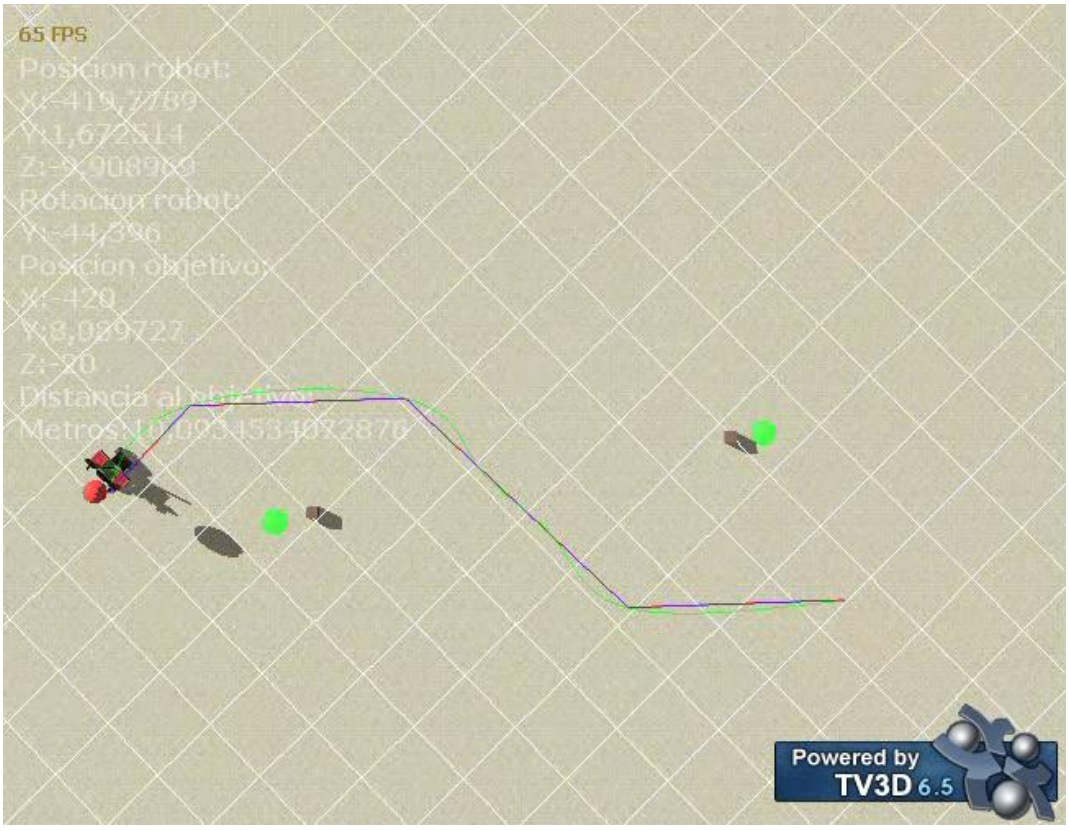


Ilustración 73: Prueba 4. Situación en el simulador al final de la ruta.

10.5.-Prueba 5: Trayectoria con algoritmo del otro grupo

Objetivo: Incluir una dll con algoritmos de análisis de visión estereoscópica para la detección de objetos.

Modo de ejecución: Ambos con visión.

Dificultad: adaptación del software al uso de una dll externa de detección de obstáculos.

El software está preparado para incluir un buen algoritmo de reconocimiento de obstáculos, el software sólo necesita la componente x del objeto y la disparidad para situar el obstáculo detectado por el algoritmo. El procedimiento de ejecución es el mismo que en la prueba anterior, la única diferencia es el algoritmo utilizado para la detección de obstáculos en las fotografías.

Resultados de la prueba nº 5			
Detalles de ejecución			
Modo de ejecución:	Ambos con visión		
GPS activado:	No		
Algoritmo de detección de obstáculos:	Algoritmo del otro grupo		
Alcanza el objetivo:	Sí		
Descripción textual			
Colocamos enfrente del robot dos obstáculos, cada uno representado por una bola amarilla. La primera está encima de una caja que oculta la segunda al principio. Así el robot, verá primero una de ellas y luego la otra. Esto hará que replanifique en dos ocasiones. Ponemos la posición objetivo de la ruta detrás del último obstáculo.			
Datos de la ruta			
Posición origen:	(X: 0, Z: 0)		
Rotación inicial:	179,9853		
Posición objetivo:	(X: -290, Z: 10)		
Distancia al objetivo (distancia simulador):	290,17		
Lista de pasos original			
Paso 1			
Nº de acción	**Tipo**	**Cantidad**	**Observaciones**
1	Girar	-2,031433	
2	Consultar giro	-	GPS no activado => No consulta
3	Visión	-	Nuevo obstáculo => Replanifica

4	Avanzar	290,1754	Replanificación => No se ejecuta
5	Consultar	-	Replanificación => No se ejecuta
6	Sincronismo	-	Replanificación => No se ejecuta
Lista de pasos tras la primera replanificación			
Paso 1			
Nº de acción	**Tipo**	**Cantidad**	**Observaciones**
1	Girar	33,69006	
2	Consultar giro	-	GPS no activado => No consulta
3	Visión	-	No replanifica
4	Avanzar	115,3777	
5	Consultar	-	GPS no activado => No consulta
Paso 2			
Nº de acción	**Tipo**	**Cantidad**	**Observaciones**
1	Girar	-33,69006	
2	Consultar giro	-	GPS no activado => No consulta
3	Visión	-	Nuevo obstáculo => Replanifica
4	Avanzar	128	Replanificación => No se ejecuta
5	Consultar	-	Replanificación => No se ejecuta
Paso 3			
Nº de acción	**Tipo**	**Cantidad**	**Observaciones**
1	Girar	-43,85519	Replanificación => No se ejecuta
2	Consultar giro	-	Replanificación => No se ejecuta
3	Visión	-	Replanificación => No se ejecuta
4	Avanzar	91,76917	Replanificación => No se ejecuta
5	Consultar	-	Replanificación => No se ejecuta
6	Sincronismo	-	Replanificación => No se ejecuta
Lista de pasos tras la segunda replanificación			
Paso 1			
Nº de acción	**Tipo**	**Cantidad**	**Observaciones**
1	Girar	11,30994	
2	Consultar giro	-	GPS no activado => No consulta

3	Visión	-	No replanifica
4	Avanzar	163,1687	
5	Consultar	-	GPS no activado => No consulta
Paso 2			
Nº de acción	**Tipo**	**Cantidad**	**Observaciones**
1	Girar	-56,30995	
2	Consultar giro	-	GPS no activado => No consulta
3	Visión	-	No replanifica
4	Avanzar	45,25483	
5	Consultar	-	GPS no activado => No consulta
Paso 3			
Nº de acción	**Tipo**	**Cantidad**	**Observaciones**
1	Girar	-43,99398	
2	Consultar giro	-	GPS no activado => No consulta
3	Visión	-	No replanifica
4	Avanzar	68,54372	
5	Consultar	-	GPS no activado => No consulta
6	Sincronismo	-	

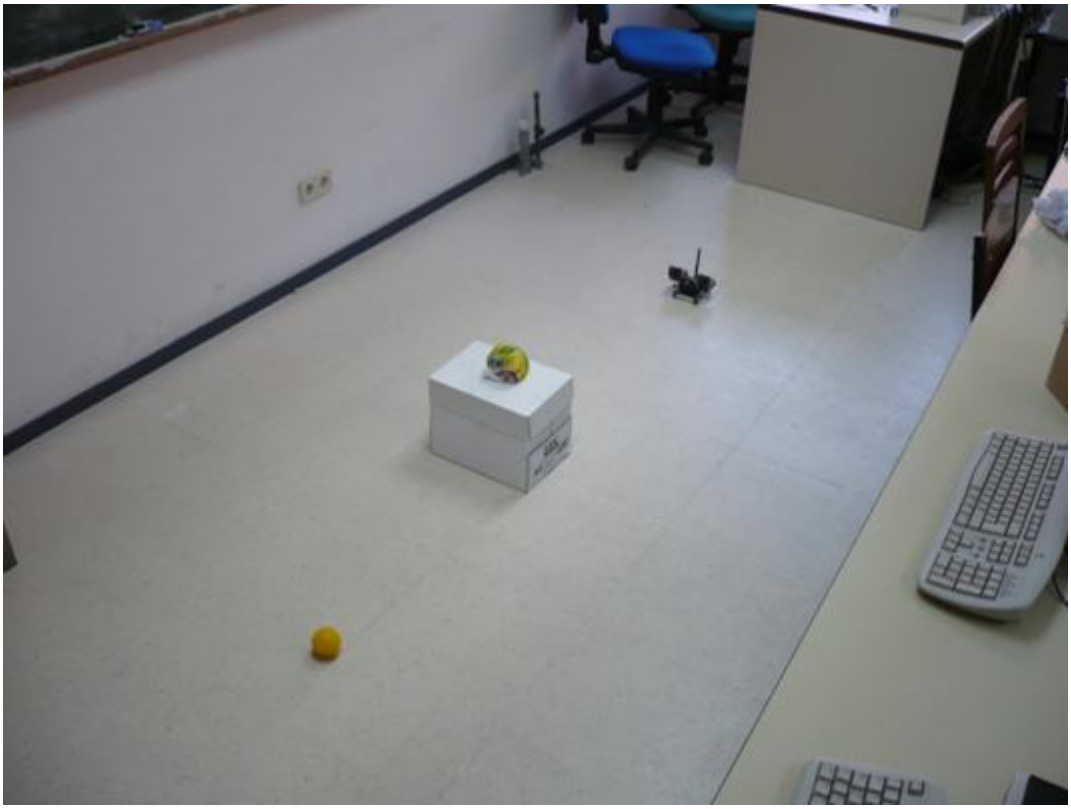


Ilustración 74: Prueba 5. Situación inicial.

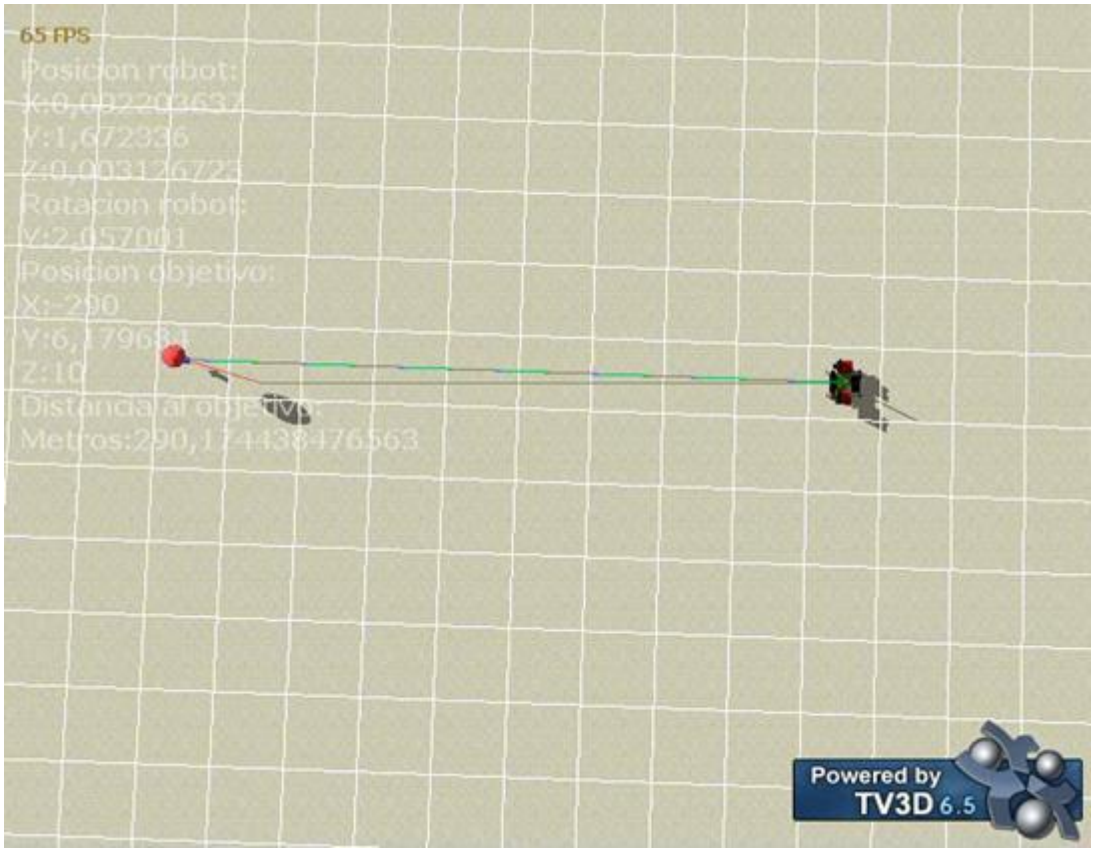


Ilustración 75: Prueba 5. Situación en el simulador al capturar las instantáneas de la acción de visión 1, inmediatamente antes de la replanificación. Observación: En la simulación la bola roja es el objetivo, ningún obstáculo ha sido encontrado por el momento y, por ello, la primera ruta calculada es una línea recta.

Ilustración 76: Prueba 5. Instantánea de la cámara izquierda en la acción de visión 1.

Ilustración 77: Prueba 5. Instantánea de la cámara derecha en la acción de visión 1.

Ilustración 78: Prueba 5. En la acción de visión 1 encuentra el primer obstáculo y replanifica.

Ilustración 79: Prueba 5. Instantánea de la cámara izquierda en la acción de visión 3.

Ilustración 80: Prueba 5. Instantánea de la cámara derecha en la acción de visión 3.

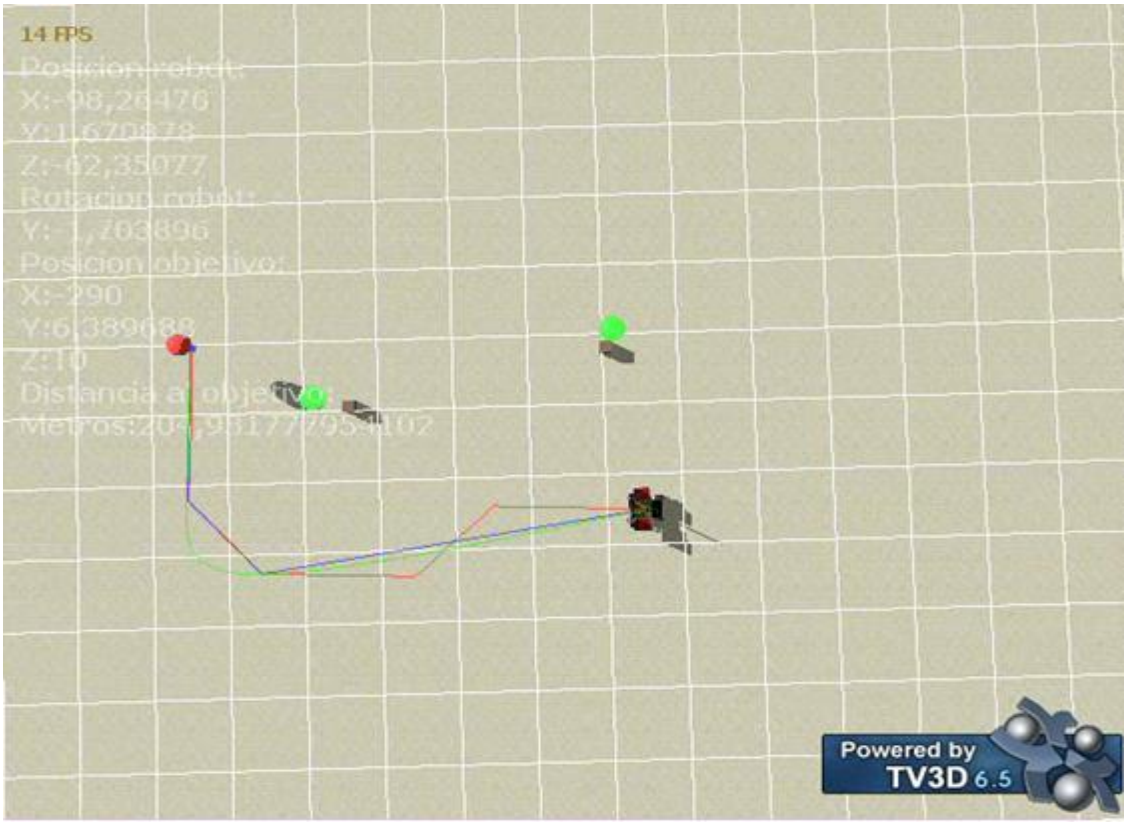


Ilustración 81: Prueba 5. En la acción de visión 3 encuentra el otro obstáculo y replanifica.

.

Ilustración 82: Prueba 5. Situación real al final de la ruta.

Ilustración 83: Prueba 5. Situación en el simulador al final de la ruta.

10.6.-Prueba 6: Trayectoria simple conociendo el entorno.

Objetivo: Planificación de una ruta con ejecución única por parte del SRV-1.

Modo de ejecución: Surveyor

Dificultad: Aislar el sistema de simulación al sistema de ejecución de pasos en SRV-1.

Es una planificación y ejecución normal pero con una ejecución sólo por parte del robot,el simulador no realiza ningún movimiento. Sólo lo planifica y por tanto no hace falta el sincronismo por parte de simulador y SRV-1

Resultados de la prueba nº 6			
Detalles de ejecución			
Modo de ejecución:	Surveyor		
GPS activado:	No		
Algoritmo de detección de obstáculos:	Ninguno. Surveyor => Sin visión		
Alcanza el objetivo:	Sí		
Descripción textual			
El robot debe llegar al objetivo pasando entre medias de dos obstáculos representados por dos cajas.			
Datos de la ruta			
Posición origen:	(X: 0, Z: 0)		
Rotación inicial:	179,7837		
Posición objetivo:	(X: -330, Z: -40)		
Distancia al objetivo (distancia simulador):	332,4		
Lista de pasos original			
Paso 1			
Nº de acción	**Tipo**	**Cantidad**	**Observaciones**
1	Girar	11,75433	
2	Consultar giro	-	GPS no activado => No consulta
3	Avanzar	163,1686	
4	Consultar	-	GPS no activado => No consulta
Paso 2			
Nº de acción	**Tipo**	**Cantidad**	**Observaciones**
1	Girar	-9,053314	
2	Consultar giro	-	GPS no activado => No

			consulta
3	Avanzar	170,28	
4	Consultar	-	GPS no activado => No consulta

Ilustración 84: Prueba 6. Situación inicial.

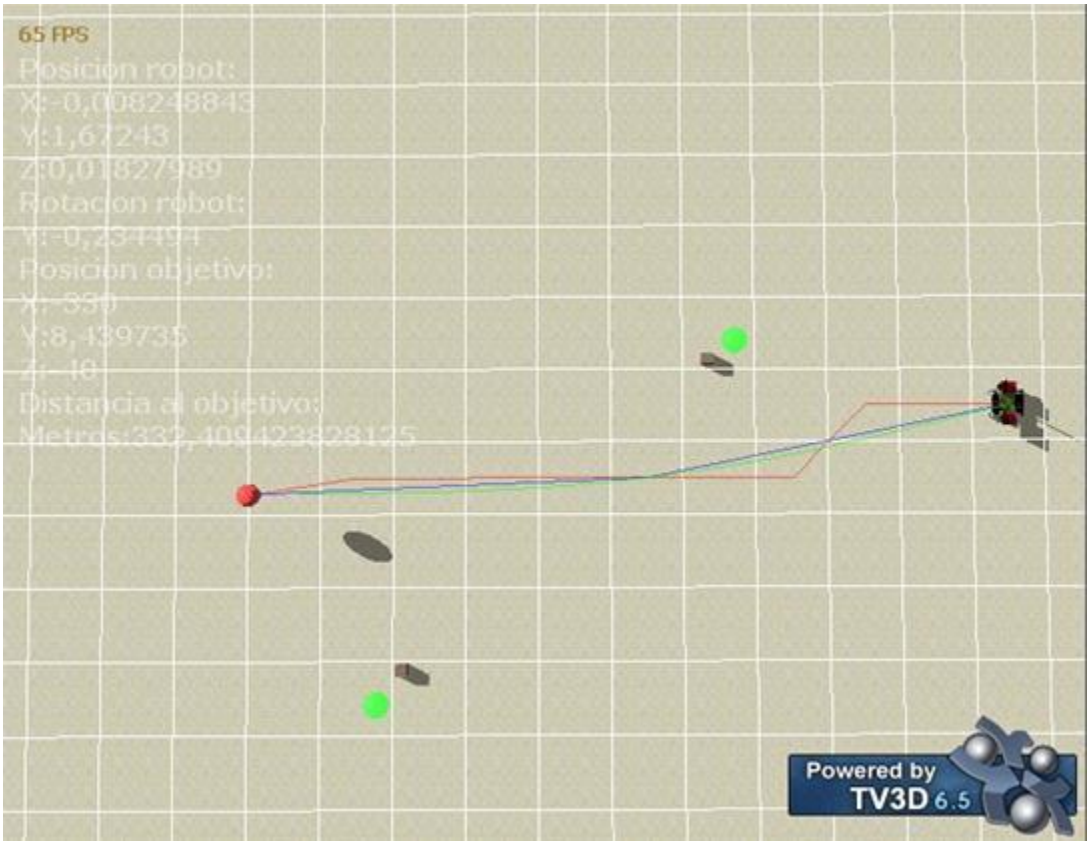


Ilustración 85: Prueba 6. Ruta planificada.

Ilustración 86: Prueba 6. Fin del primer paso de la ruta.

Ilustración 87: Prueba 6. Fin de la ruta.

10.7.-Prueba 7: Trayectoria inviable

Objetivo: Parar ante destinos inviables por peligro de colisión.

Modo de ejecución: Ambos con visión

Dificultad: Detección de una situación peligrosa para el robot, detectamos que la coordenada de destino es un sitio inalcanzable puesto que hay peligro de colisión con un objeto que hemos situado por la visión estereoscópica.

Situamos un destino y cuando nos acercamos a ese destino descubrimos que hay un obstáculo en esa coordenada o cercano a ella. Esto podemos hacerlo gracias al modo ambos con visión el cual nos hace evitar esa situación de peligro y nos para la ejecución de la trayectoria. Seguidamente podríamos insertar otra coordenada como destino.

Resultados de la prueba nº 7			
Detalles de ejecución			
Modo de ejecución:	Ambos con visión		
GPS activado:	No		
Algoritmo de detección de obstáculos:	Algoritmo basado en colores		
Alcanza el objetivo:	No		
Descripción textual			
Prueba de un camino inviable. Hay dos obstáculos. El primero no es problemático, pero el segundo impide alcanzar el objetivo. La ejecución de la ruta se aborta.			
Datos de la ruta			
Posición origen:	(X: 0, Z: 0)		
Rotación inicial:	179,9853		
Posición objetivo:	(X: -240, Z: 0)		
Distancia al objetivo (distancia simulador):	239,96		
Lista de pasos original			
Paso 1			
Nº de acción	**Tipo**	**Cantidad**	**Observaciones**
1	Girar	-0,08355713	
2	Consultar giro	-	GPS no activado => No consulta
3	Visión	-	Nuevo obstáculo => Replanifica
4	Avanzar	239,9717	Replanificación => No se ejecuta

5	Consultar	-	Replanificación => No se ejecuta
6	Sincronismo	-	Replanificación => No se ejecuta
Lista de pasos tras la primera replanificación			
Paso 1			
Nº de acción	**Tipo**	**Cantidad**	**Observaciones**
1	Girar	33,69008	
2	Consultar giro	-	GPS no activado => No consulta
3	Visión	-	No replanifica
4	Avanzar	115,3777	
5	Consultar	-	GPS no activado => No consulta
Paso 2			
Nº de acción	**Tipo**	**Cantidad**	**Observaciones**
1	Girar	-33,69008	
2	Consultar giro	-	GPS no activado => No consulta
3	Visión	-	Nuevo obstáculo => Replanificación => Ruta inviable
4	Avanzar	63,99999	Ruta inviable => No se ejecuta
5	Consultar	-	Ruta inviable => No se ejecuta
Paso 3			
Nº de acción	**Tipo**	**Cantidad**	**Observaciones**
1	Girar	-45	Ruta inviable => No se ejecuta
2	Consultar giro	-	Ruta inviable => No se ejecuta
3	Visión	-	Ruta inviable => No se ejecuta
4	Avanzar	45,25484	Ruta inviable => No se ejecuta
5	Consultar	-	Ruta inviable => No se ejecuta
Paso 4			
Nº de acción	**Tipo**	**Cantidad**	**Observaciones**
1	Girar	11,00433	Ruta inviable => No se ejecuta
2	Consultar giro	-	Ruta inviable => No se ejecuta
3	Visión	-	Ruta inviable => No se ejecuta
4	Avanzar	57,85799	Ruta inviable => No se ejecuta
5	Consultar	-	Ruta inviable => No se ejecuta
6	Sincronismo	-	Ruta inviable => No se ejecuta

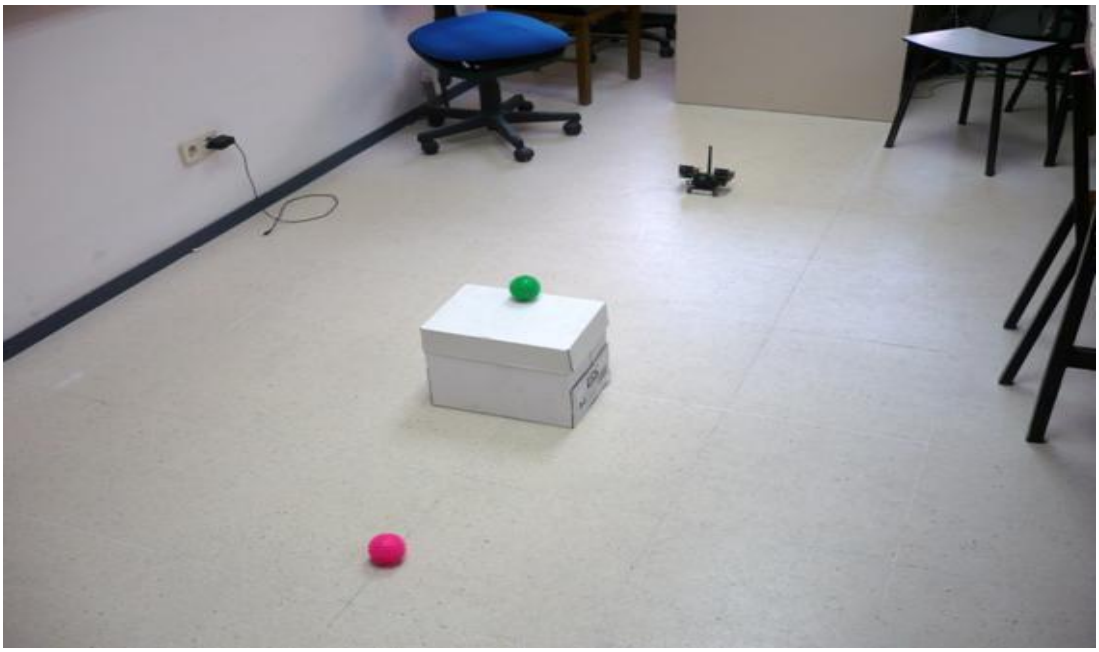


Ilustración 88: Prueba 7. Situación inicial.

Ilustración 89: Prueba 7. Instantánea de la cámara izquierda en la acción de visión 1.

Ilustración 90: Prueba 7. Instantánea de la cámara derecha en la acción de visión 1.

Ilustración 91: Prueba 7. En la acción de visión 1 encuentra el primer obstáculo y replanifica.

Ilustración 92: Prueba 7. Instantánea de la cámara izquierda en la acción de visión 3.

Ilustración 93: Prueba 7. Instantánea de la cámara derecha en la acción de visión 3.

Ilustración 94: Prueba 7. Mensaje de error y fin de la ruta. Obsérvese que el objetivo está muy próximo al segundo obstáculo. Esa es la razón de que la ruta sea inviable.

11.-Anexos

11.1.-TRATAMIENTO DE IMÁGENES CON METODOS UNSAFE

Debido a la necesidad de realizar un tratamiento de las imágenes de forma rápida y eficiente en el entorno .net utilizar las librería grafica propia GDI+ no es recomendable, así que se utilizan métodos UNSAFE para tal labor, obteniendo unos resultados aceptables de acuerdo a las necesidades del proyecto.

El modo UNSAFE permite acceder a esta parte del código sin utilizar el CLR(Common Language Runtime) lo que permite una eficiencia necesaria para nuestro propósito, para ello además de indicarlo en el código es necesario marcar en el entorno de desarrollo que el código inseguro este permitido.

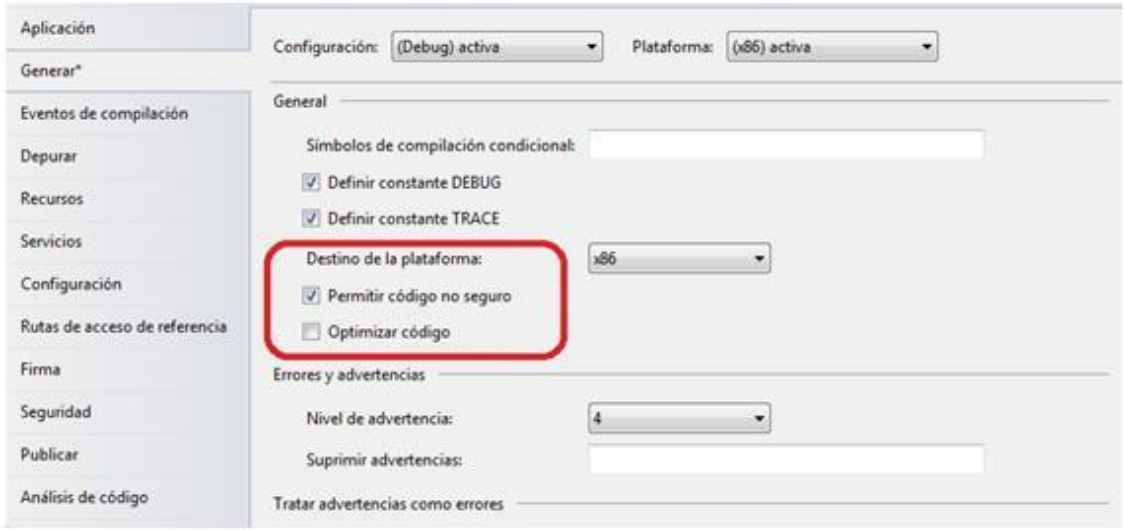


Ilustración 95: Captura del IDE con el objetivo de insertar código inseguro

En el código a continuación se muestra además del método unsafe el tratamiento de imágenes que realizamos basándonos en colores, la dificultad no radica en el código sino mas bien en los limites que imponemos para considerar que se encuentran obstáculos de colores rojo verde o azul, muy dependientes de la iluminación de las escenas.

Código para encontrar objetos de colores:

```
public static void posicionPuntoImagenUnsafe1(Bitmap b)
        {

            bmData = b.LockBits(new Rectangle(0, 0, b.Width,
b.Height), ImageLockMode.ReadOnly, PixelFormat.Format24bppRgb);
            Scan0 = bmData.Scan0;
            int cr = 0, cg = 0, cb = 0;
            int crr = 0, xr = 0, yr = 0;
            int crg = 0, xg = 0, yg = 0;
```

```
        int crb = 0, xb = 0, yb = 0;
        unsafe
        {
            byte* p = (byte*)(void*)Scan0;
            for (int y = 0; y < b.Height; ++y)
            {
                for (int x = 0; x < b.Width; ++x)
                {
                    cb = p[0];
                    ++p;
                    cg = p[0];
                    ++p;
                    cr = p[0];
                    ++p;
                    if (cr - cg > 180 & cr - cb > 180)
                    {
                        xr = xr + y;
                        yr = yr + x;
                        crr++;
                    }
                    if (cg - cr > 140 & cg - cb > 83)
                    {
                        xg = xg + y;
                        yg = yg + x;
                        crg++;
                    }
                    if (cr>230 & cg>210 & cb<140)
                    {
                        xb = xb + y;
                        yb = yb + x;
                        crb++;
                    }
                }
            }
        }
        if (crr == 0)
            crr = 10000;
        if (crg == 0)
            crg = 10000;
        if (crb == 0)
            crb = 10000;
        pospun1r.x = yr / crr;
        pospun1r.y = xr / crr;
        pospun1g.x = yg / crg;
        pospun1g.y = xg / crg;
        pospun1b.x = yb / crb;
        pospun1b.y = xb / crb;
        b.UnlockBits(bmData);
    }
```

Se resume en un recorrido por cada pixel de la imagen contabilizando si es rojo, verde o azul para a posteriori sacar la media de esos puntos, una vez tenemos el punto central de un objeto en una de las imágenes se hace lo mismo con la imagen de la otra cámara y con la explicación del siguiente anexo sacamos la posición del objeto.

11.2.-POSICIONAMIENTO DE UN OBJETO EN UN ENTORNO 3D

Ángulo del Objeto respecto al robot

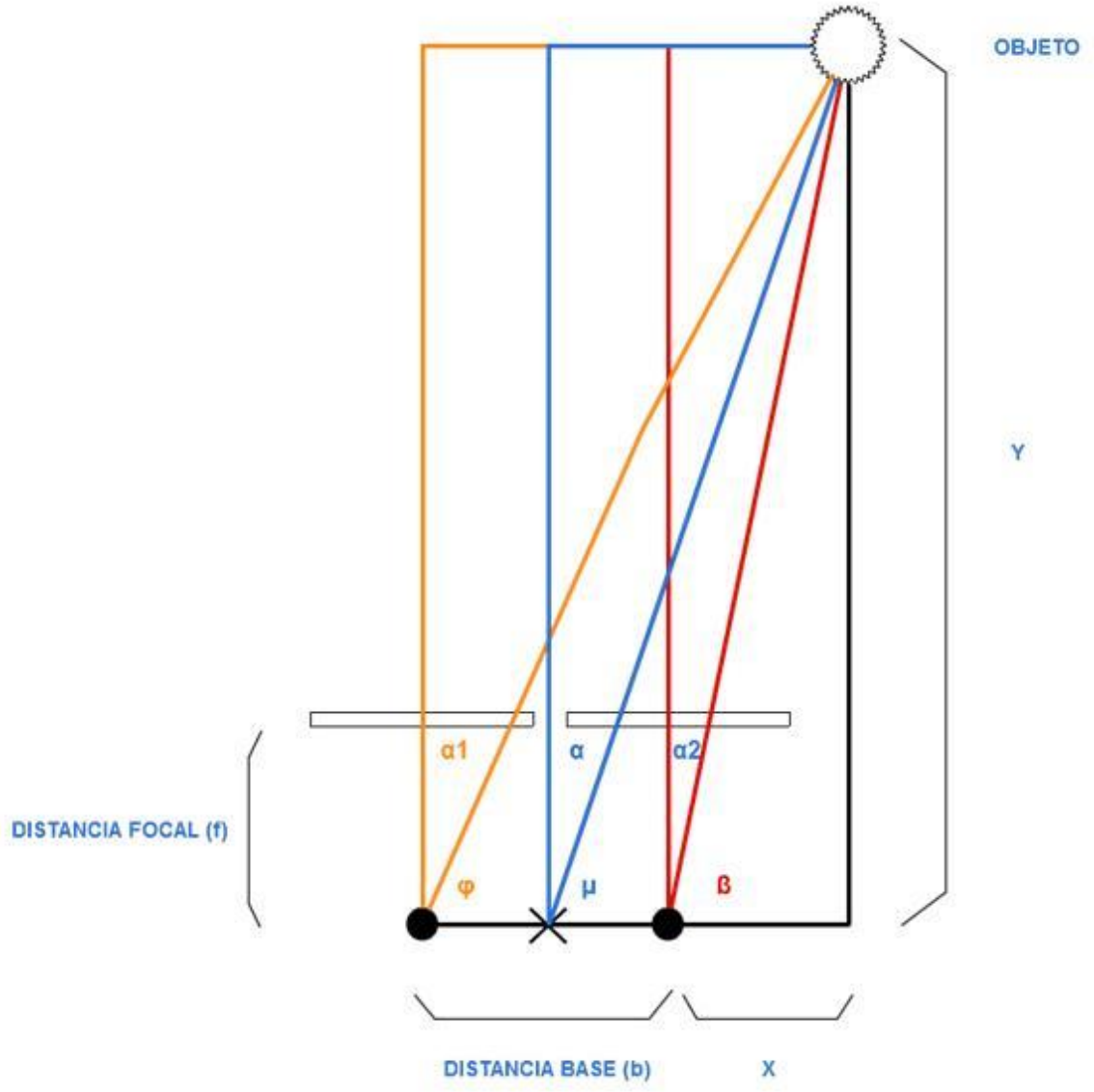


Ilustración 96: Esquema de la visión estéreo

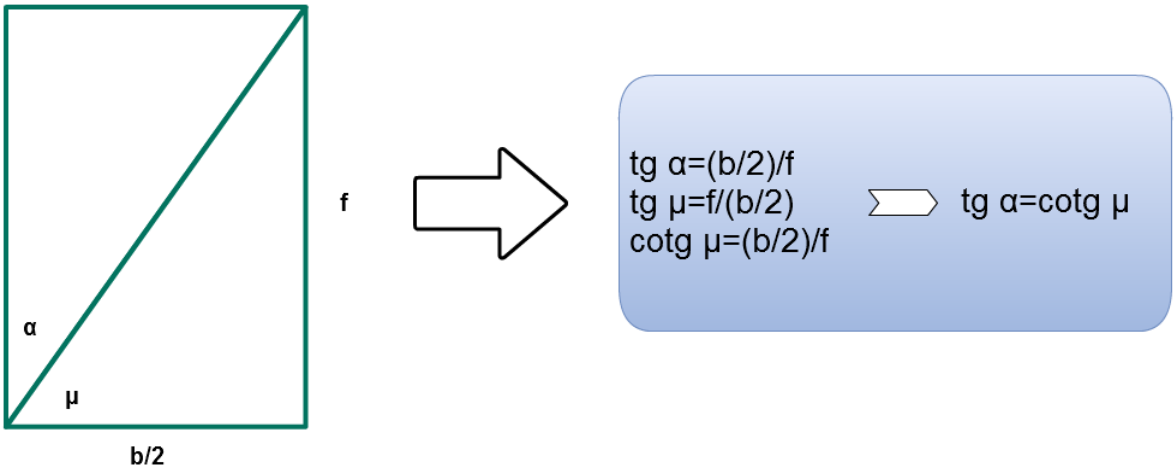


Ilustración 97: Trigonometría básica utilizada

Generalizando

$$tg\ (90 - \alpha) = tg\ (\mu) = cotg\ \alpha$$

Si aplicamos esto en la anterior formula nos queda

$$\frac{tg\ \alpha_2 + tg\ \alpha_1}{2} = tg\ \alpha$$

Si tomamos en cuenta los ángulos que hace respecto a las lentes tenemos:

$$tg\ \alpha = \frac{\frac{x_1}{f} + \frac{x_2}{f}}{2}$$

*donde x_1 y x_2 son la posición x del objeto en la imagen izquierda y derecha.

Lo que simplificando llegamos:

$$tg\ \alpha = \frac{x_1 + x_2}{2f}$$

Con lo que

$$\alpha = arcotg(\frac{x_1 + x_2}{2f})$$

Éste ángulo α será positivo si el objeto está a la derecha y negativo si está a la izquierda y el valor de éste estará entre [-90,90].

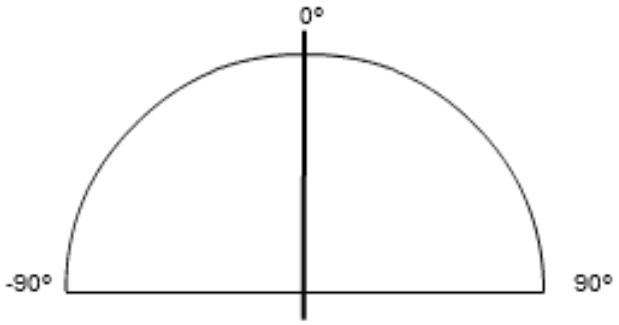


Ilustración 98: Grados de giro del obstáculo en el posicionamiento tridimensional.

Tenemos el ángulo del objeto respecto al robot, si con éste ángulo unimos el que tiene el robot y la distancia (d) que podemos hallar a partir de la disparidad, entonces obtendríamos unas coordenadas polares.

$$rot_{total} = rot_{robot} + \alpha$$

$$x = d \cos (rot_{total})$$

$$y = d\ sen\ (rot_{total})$$

Hallando esas coordenadas en el plano (x,z) entonces obtendremos la posición del objeto respecto al robot.

Con estas coordenadas le sumamos la posición del robot nos colocará el objeto respecto al robot en el entorno 3D.

$$x_{objeto} = x_{robot} + x$$

$$y_{objeto} = y_{robot} + y$$

Hemos de decir, que en nuestra simulación está preparada para tener señal de GPS, esto nos llevaría a obtener unas coordenadas con precisión. Debido a que no disponemos de GPS creamos una estimación y esto nos puede llevar a que no obtengamos la posición del objeto con una precisión total.

11.3.-SINCRONIZACION MEDIANTE UN TIMER

Existen tres tipos de controles de temporizador en Visual Studio y .NET Framework: el temporizador basado en servidor, que puede verse en la ficha **Componentes** del **Cuadro de herramientas**; el temporizador basado en Windows estándar, que puede verse en la ficha **Windows Forms** del **Cuadro de herramientas**, y el temporizador de subprocesos, que únicamente está disponible mediante programación.

Los temporizadores se diseñan con diferentes fines, como indica su control de subprocesos:

- El temporizador de Windows está diseñado para un entorno de un solo subproceso, donde se utilizan subprocesos de interfaz de usuario para la ejecución de los procesos. La precisión de los temporizadores de Windows está limitada a 55 milisegundos. Estos temporizadores tradicionales necesitan que el código del usuario tenga un suministro de mensajes de interfaz de usuario y que funcione siempre desde el mismo subproceso o calcule las referencias de la llamada a otro subproceso. Para un componente COM, esto perjudicaría al rendimiento.
- El temporizador basado en servidor está diseñado para utilizarse con subprocesos de trabajo en un entorno de múltiples subprocesos. Puesto que utilizan una arquitectura diferente, los temporizadores basados en servidor pueden ser mucho más precisos que los temporizadores de Windows. Los temporizadores de servidor pueden moverse entre los subprocesos para controlar los eventos que se produzcan.
- El temporizador de subprocesos es útil en escenarios donde no se bombean mensajes en el subproceso. Por ejemplo, el temporizador basado en Windows depende de la compatibilidad con temporizador del sistema operativo y, si no se bombean mensajes en el subproceso, no se produce el evento asociado con el temporizador. El temporizador de subprocesos resulta más útil en este caso.

El componente **Timer** es un temporizador basado en servidor que permite especificar un intervalo recurrente en el que se provoca el evento Elapsed en la aplicación. Entonces, se puede controlar este evento para proporcionar un procesamiento normal.

El **Timer** basado en servidor está diseñado para utilizarlo con subprocesos de trabajo en un entorno multiproceso. Los temporizadores basados en servidor pueden desplazarse entre subprocesos para controlar el evento **Elapsed** provocado, dando lugar a una mayor precisión que la proporcionada por los temporizadores de Windows al provocar el evento en el momento exacto.

El componente **Timer** provoca el evento **Elapsed**, que se basa en el valor de la propiedad Interval. Se puede controlar este evento para realizar el procesamiento que se necesite.

Este componente se puede considerar el "motor" de nuestro software. En él tendremos el cálculo de la física y renderización del simulador. Por medio de éste controlamos los movimientos del simulador, de forma que hacemos que funcione igual en un equipo que en otro, puesto que si no fuera así tendríamos problemas de avances y giros con el simulador.

Esta llamada es la que se ejecuta en cada evento del timer:

```
private void timer1_Tick(object sender, EventArgs e)
```

Todo lo que hay dentro de esta función se ejecutará cada 5 milisegundos que es el valor que hemos definido en la propiedad Interval del timer.

```
motor.Run();
motor._tv.RenderToScreen();
```

Estas dos funciones se encargan del cálculo de la física y de renderizar el motor gráfico en pantalla en cada vuelta del timer.

Como hemos comentado anteriormente, el timer es el motor de nuestro software y por ello vamos a explicar el funcionamiento que lleva a cabo. Una de las cosas más importantes que maneja son los movimientos del robot de la simulación. Veamos cómo funciona los movimientos del robot en la simulación.

Para realizar movimientos que avancen lo mismo tuvimos que forzar a la física que se recalculase cada cierto tiempo fijo, puesto que al ser variable la física cambiaba según las caracteríticas del computador usado. Teniendo esto ya fijado, el siguiente paso era ejecutarlo en ciertos tiempos iguales,la solución es el uso de la componente timer.

Las funciones de movimiento del robot de simulación reciben como parametro una distancia o grado, lo que convertimos en el concepto de ticks. Estos ticks es una conversión para establecer una relación entre timer-avance o timer-grados. Como salida de estas funciones son variables booleanas indicando si ha terminado su movimiento(true) o no(false).

Dentro de estas funciones de movimiento su funcionamiento es simple:

1. Se pide el movimiento y se lanza a ejecutar por primera vez, la variable ticks recibe el valor de la conversión de grados o distancia a ticks y la variable tick_interno es un contador que inicializamos a 0, cuando esto se produce invocamos al robot a moverse,y la función delvolverá false(movimiento no terminado).
2. En cada vuelta del timer ,mientras que no hayamos terminado el movimiento,ejecutaremos esta función y el contador tick_interno se irá

incrementando hasta que llegue al valor de la conversión.La función sigue devolviendo false.

3. En este momento se le manda al robot parar el movimiento pero la función seguirá devolviendo false hasta que el contador sea un poco más grande para que la inercia de la física se complete.
4. Una vez se llegue a un valor de tick_interno mas alto (pre-establecido anteriormente) la función de movimiento devolverá true(movimiento terminado) e inicializamos el contador interno (tick_interno).

El problema del control de la inercia surgió cuando quisimos enlazar movimientos,si ejecutamos una lista de movimientos consecutivos sin un control de inercia lo que nos da como resultado son movimientos más cortos, puesto que se enlazan sin haber terminado el movimiento completamente,es decir, sin contar con la inercia del motor de física.

Esta es la función del avance del robot de la simulación, perteneciente a la clase Robot.cs

```
public bool mover_frente(float distancia)
        {
            int tick = convDistTick(distancia);     //CONVERSION DISTANCIA A TICKS
            if (tick_interno == 0)                  //Inicio del movimiento
            {
                CarPower = 1000;                    //Potencia de las ruedas
                Accelerateleftwheelsup();           //Activar ruedas izquierdas
                Acceleraterightwheelsup();          //Activar ruedas derechas
                tick_interno = tick_interno + 1;    //incremento del contador interno
                return false;
            }
            if (tick_interno < tick)                //Incremento del contador interno
            {
                tick_interno = tick_interno + 1;
                return false;
            }
            else if (tick_interno == tick)          //Hemos llegado al valor de la
            {                                       //conversion,parar robot
                Physics.SetVehicleWheelTorque(_idrobot, _idruedadd, 0, CarDamp);
                Physics.SetVehicleWheelTorque(_idrobot, _idruedadi, 0, CarDamp);
                Physics.SetVehicleWheelTorque(_idrobot, _idruedati, 0, CarDamp);
                Physics.SetVehicleWheelTorque(_idrobot, _idruedatd, 0, CarDamp);
                tick_interno = tick_interno + 1;
                return false;
            }
            else if (tick_interno > tick)           //Controlamos la inercia de la
            {                                       //fisica
                tick_interno = tick_interno + 1;
                if (tick_interno > tick + 150)      //Fin del control de inercia.
                {
                    tick_interno = 0;               //Inicializamos el contador
interno.
                    return true;
                }
                return false;
            }
            else
                return false;
           }
```

El `timer1_Tick` se va complicando según avanza el proyecto. Surgen nuevas necesidades de aumentarlo para el control de todo el simulador y del SRV-1. Alrededor de éste se controla los pasos que se realizan de la lista de pasos obtenida del cálculo de la ruta y distingue los modos para realizar unas comprobaciones u otras según el modo de ejecución (modo simulador, Surveyor, ambos, ambos con visión…) .

CONCLUSIONES

Los objetivos marcados en un principio han sido logrados de una manera satisfactoria. La metodología y la organización utilizada por el grupo nos ha permitido poder superar las dificultades encontradas y poder adaptarnos a las necesidades del proyecto.

Para comenzar el trabajo ha sido necesaria una introducción en ciertos campos que desconocíamos. Gracias a ello, hemos tenido la posibilidad de descubrir los distintos planteamientos y posibilidades que ofrecen este tipo de aplicaciones y herramientas.

Cabe destacar, que este proyecto nos ha aportado conocimientos técnicos sobre el campo de la robótica, inteligencia artificial e informática gráfica. Adicionalmente nos ha formado en el desarrollo de software bajo las premisas del trabajo en grupo de manera coordinada, apoyándonos unos a otros para el desarrollo completo de una herramienta de estas características.

A lo largo de éste proyecto hemos aprendido las dificultades que surgen al diseñar e implementar un software especifico para un hardware concreto. Éste hardware con sus propias restricciones impuestas por un entorno real como puede ser la sincronización, toma de decisiones, precisión… nos llevará a tener conocimiento general de algunos campos y más específico de otros.

En cuanto a tendencias futuras de este software, hay grandes alternativas, un siguiente avance de este software podría ser el tratamiento sobre terrenos irregulares. Este tipo de tendencias pueden llegar a ser un avance en campos como la tripulación de robots terrestres en otros planetas o robots para desactivación bombas.

La herramienta con un algoritmo correcto de reconocimiento de obstáculos y un sistema de posicionamiento global podría dar lugar a un sistema completo de navegación no tripulado adecuado para cualquier tipo de propósito.

BIBLIOGRAFÍA

FU,K.S., González, R.C. y Lee, G.S.G. *Robótica, Control, Detección, Visión e Inteligencia*. Mc Graw-Hill, 1990. ISBN: 9789684223578

PRESSMAN, Roger S. *Software Engineering: a practitioner's approach*. 6ª ed. Mc Graw-Hill, 2005. ISBN: 9780073019338

SIEGWART R. and Nourbakhsh I. R. *Introduction to autonomous mobile robots*. MIT Press, 2004. ISBN: 9780262195027

AFORGE.NET, página web [en línea] <www.aforgenet.com/framework>

AUTODESK MAYA, página web [en línea] <http://usa.autodesk.com/adsk/servlet/pc/index?id=13577897&siteID=123112>

MICROSOFT VISUAL STUDIO, MSDN, página web [en línea] <http://msdn.microsoft.com/es-es/vstudio/default.aspx>

TRUEVISION3D, página web [en línea] <http://www.truevision3d.com/>

TV3DWIKI, página web [en línea] <vhttp://wiki.truevision3d.com/>

WIKIPEDIA. *Newton Game Dynamics* [en línea] <http://en.wikipedia.org/wiki/Newton_Game_Dynamics>

WIKIPEDIA. *Physics Engine* [en línea] < http://en.wikipedia.org/wiki/Physics_engine>

WIKIPEDIA. *Mobile Robot* [en línea] <http://en.wikipedia.org/wiki/Mobile_robot>

NASA. *Mars Exploration Rover Mission* [en línea] <http://marsrover.nasa.gov/home/>

HONDA. *The Honda Humanoid Robot ASIMO* [en línea] <http://world.honda.com/ASIMO/>

AUTOMOWER, página web [en línea] <http://www.automower.com/node2923.aspx?nid=139624&pid=88560>

SONY AIBO EUROPE, página web [en línea] <http://www.eu.aibo.com>

FOCAL-LENGTH-FOV CALCULATOR, página web [en línea] <http://kmp.bdimitrov.de/technology/fov.html>