Capítulo 2

Vehículo Autónomo Romeo-4R[1]

2.1. Introducción

Romeo-4R es un robot móvil que viene siendo desarrollado en el Departamento de Ingeniería de Sistemas y Automática de la Escuela Superior de Ingenieros de Sevilla desde la década de los 90.

Su nombre deriva de *Robot Móvil para Exteriores*, y es el resultado de la adaptación de un vehículo eléctrico convencional de cuatro ruedas (4R). Este tipo de vehículos se utilizaron durante la Exposición Universal de Sevilla de 1992 y fue adquirido por el Departamento de Ingeniería de Sistemas y Automática tras la finalización de la misma. Tal y como puede verse en la figura 2.1, es el equivalente a un antiguo carrito de golf.

La principal función de Romeo-4R es la experimentación de distintas técnicas de navegación en exteriores, mediante la utilización de los sensores y actuadores de los que dispone. A día de hoy, aún se puede admirar en los laboratorios a su predecesor, Romeo-3R. Pero en adelante y para el resto de este texto, cuando se hable de «Romeo», se deberá entender como una referencia a Romeo-4R.

En lo relativo a los sistemas de referencia que se utilizarán a lo largo del presente proyecto, hay que distinguir entre el sistema de coordenadas global o del mundo (WCS) y el sistema de coordenadas local solidario al robot (RCS). La utilización de al menos estos dos sistemas de referencias es muy habitual en los sistemas robóticos móviles.

Para el caso de Romeo-4R, con origen en la proyección del punto medio del eje trasero en el plano del suelo, el sistema RCS se compone de forma que el eje y se dirige hacia adelante, el eje z hacia el cielo, y el eje x de manera que resulte un sistema dextrógiro, como podemos ver en la figura 2.2.

[1] Este capítulo está basado en el trabajo [24]

Figura 2.1: Robot Romeo-4R.

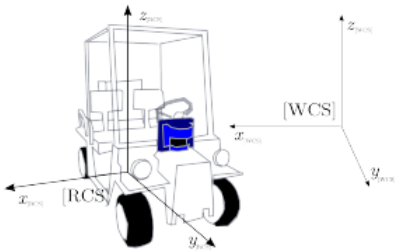

Figura 2.2: Sistemas de referencia global y local.

Aunque este tipo de definición es la mas completa, suele utilizarse poco en la robótica terrestre, donde la altura del robot viene determinada por la altura del suelo que pisa. Por ello es más habitual en estos casos utilizar un sistema de referencia plano, proyección del anterior, ver figura 2.3. En él existe un punto del mapa que se considera origen del sistema de coordenadas global, y unos ejes definidos. Como sólido rígido en movimiento en el plano, el robot dispone de tres grados de libertad que quedan definidos por la posición en el sistema global del origen de su sistema local (x, y) y por la orientación de sus ejes respecto a los globales (θ).

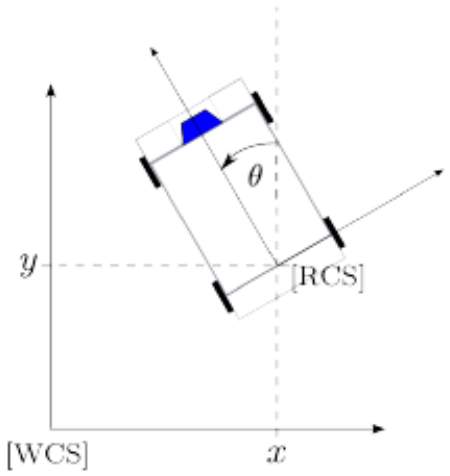

Figura 2.3: Particularización de los sistemas de referencia para el caso plano.

2.2. Equipamiento Hardware

Con unas dimensiones que rondan los 2,80m x 1,40m x 2,10m y un peso aproximado de 700 kg, podría decirse que el soporte físico de Romeo es un vehículo de cuatro ruedas convencional.

El sistema de tracción dispone de dos ruedas paralelas motrices colocadas sobre el eje transversal trasero del vehículo. Estas ruedas son movidas por un motor de corriente continua de 36 V de gran potencia (2 CV) que le permite alcanzar velocidades de hasta 3 m/s. El sistema de dirección está formado por dos ruedas directrices unidas mediante un eje rígido, utilizándose otro motor de corriente continua para controlar la dirección.

La rueda delantera interna gira un ángulo ligeramente superior a la externa para evitar el deslizamiento. Las prolongaciones de los ejes de las dos ruedas delanteras intersectan en un punto sobre la prolongación del eje de las ruedas traseras. El lugar de los puntos de contacto de cada rueda delantera con el suelo forma dos circunferencias aproximadamente concéntricas, resultando válido el modelo cinemático de la bicicleta, en la descripción de su movimiento en coordenadas globales. En estas ecuaciones x, y, θ son la posición y la orientación del vehículo respectivamente, ν es la velocidad longitudinal y γ la curvatura, o lo que es lo mismo, la inversa del radio de giro que se describirá para un ángulo concreto en las ruedas de dirección.

$$\dot{x} = -\nu \cdot \sin\theta \tag{2.1}$$

$$\dot{y} = \nu \cdot \cos\theta \tag{2.2}$$

$$\theta = \nu \cdot \gamma \tag{2.3}$$

Diseñado para que sea posible su conducción tanto manual como automática, durante el funcionamiento manual el conductor puede mover libremente el volante, acelerar y frenar, mientras que en el modo automático se bloquea el volante y el acelerador se anula.

Como controladores, monta un par de PCs industriales, uno dedicado al control y otro a la visión. Para los trabajos aquí desarrollados se ha empleado el de control, que cuenta con un procesador Intel Pentium 4 a 2,40GHz sobre el que corre GNU/Linux Debian 2.6.18. Este PC dispone de una tarjeta de control de motores DCX-PC 100, de la marca Precision MicroControl Corporation. Como indica su nombre se utiliza para el control de los motores de tracción y dirección, es decir, sirve de interfaz entre el ordenador y el propio hardware de control de motores. Consta de dos módulos MC-200 que no son más que servocontroladores de motores de corriente continua. Hay un módulo para cada motor donde cada uno utiliza un control PID programable con realimentación directa de la lectura de cada codificador. Además de estos codificadores (que permiten generar medidas de odometría) Romeo dispone de una serie de sensores y elementos que le permiten realizar medidas del entorno y aumentar su utilidad, como puede verse en las figuras 2.4 y 2.5, y que pasamos a describir a continuación.

- **Giróscopo**

  El modelo utilizado, un Autogiro Navigator Plus de KVH Industries, está construido como un interferómetro de fibra óptica de un solo eje y resulta muy adecuado para sistemas de navegación terrestre. Proporciona medidas de la velocidad angular de giro respecto al eje z del robot, perpendicular al plano del suelo. La integración de dichas medidas en el tiempo permite estimar el giro del vehículo, y calcular as la orientación del mismo. El giróscopo se comunica a través de puerto serie a una velocidad de 9600 Bd.

- **IMU**

  La unidad de medida inercial a bordo de Romeo es en realidad un compás y magnetómetro modelo EZ-COMPASS-3A de Advanced Orientation Systems Inc. Comunica su orientación 3D a través de puerto serie con una tasa de actualización de 10Hz y una resolución de 0,08°.

Figura 2.4: Robot Romeo-4R. Sensores (I).

Figura 2.5: Robot Romeo-4R. Sensores (II).

- **GPS**

  Un receptor Novatel OEM conectado a través de un puerto serie permite la estimación de la posición global del robot en aquellos lugares con cobertura GPS.

  Para conseguir una mayor precisión en las medidas puede utilizarse en modo diferencial (DGPS), en cuyo caso habrá dos dispositivos GPS, uno montado en el robot que actuará de estación remota y otro montado en una posición conocida actuando de estación base. Ambos dispositivos se comunicarán entre sí mediante radiomódems. En este modo de trabajo el error en la posición es de tan sólo unos pocos centímetros.

- **Láser SICK**

  Se trata de un escáner láser bidimensional de medida de distancias de no contacto basado en el tiempo de vuelo de pulsos láser. El modelo es LMS 220-30106 (versión para exteriores) de la marca SICK Optic Electronic. Es un láser bidimensional que mide distancias en el plano horizontal con ángulos de exploración (100°ó 180°) y resolución (0,25°-0,5°ó 1°) programables y capaz de medir distancias de hasta 80 m. Hay que destacar que el láser LMS 220 no necesita que los objetos tengan propiedades reflectoras especiales para detectarlos. No obstante, cuanto más reflectante sea el objeto mejor será la detección. Por ejemplo, los objetos de color blanco serán mejor detectados que los de color oscuro.

- **Láseres HOKUYO**

  Romeo cuenta además con un par de láseres modelo URG-04LX de HOKUYO. De dimensiones muy reducidas (50 x 50 x 50 mm) y gran ángulo de exploración de (240°con resolución de 0,36°), este sensor lanza a 10 Hz medidas entre 20 - 4000 mm con una precisión de 10 mm. A pesar de que están especificados para aplicaciones en interiores, el comportamiento en exteriores para el montaje actual es más que aceptable.

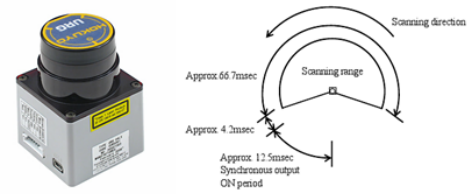

Figura 2.6: Láser Hokuyo URG-04LX.

Finalmente, se añadió un sensor láser más a Romeo, el modelo UTM-30LX de HOKUYO, instalado en el pan & tilt superior, que conocida la inclina-

ción del mismo y orientado hacia el suelo, permite la creación de mapas de elevación del entorno.

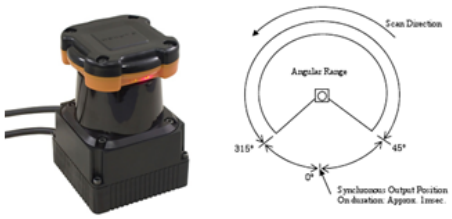

Figura 2.7: Láser Hokuyo UTM-30LX.

- **ELO Touchscreen Serie ET1515L**

  Romeo-4R cuenta con una pantalla táctil de la serie ET1515L del fabricante ELO que facilita la introducción de comandos y el manejo de la interfaz del vehículo.

- **Cámara Video DFK 21BF04**

  Romeo-4R cuenta también con una videocámara FireWire del fabricante The Imaging Source, modelo DFK 21BF04 que se utiliza para labores de reconocimiento facial y de apoyo a la navegación y localización.

Figura 2.8: Videocámara DFK 21BF04.

2.3. Arquitectura Software

La incorporación en Romeo de las aplicaciones objeto del presente trabajo ha sido posible gracias a la arquitectura software existente. Totalmente modular, en ella no se hace uso del esquema clásico cliente-servidor, sino que su modo de funcionamiento está inspirado en las redes peer-to-peer o redes de pares, donde cada uno de los nodos de la red no es cliente ni servidor fijo, sino que interactúan como iguales entre sí, algo que se adapta a las necesidades actuales de los sistemas robóticos avanzados.

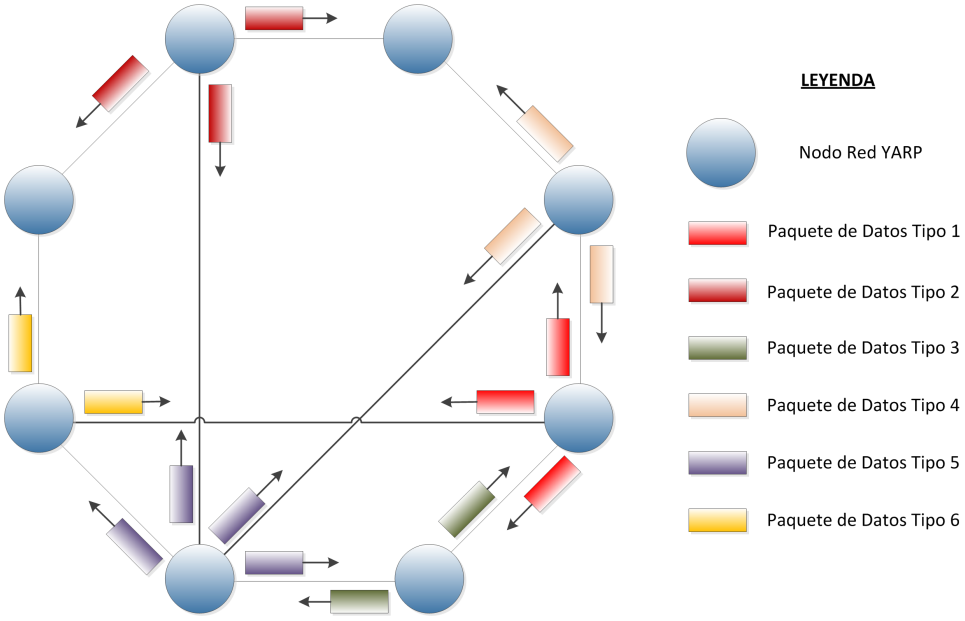

Figura 2.9: Red YARP - Modo peer to peer.

Así, cada uno de los múltiples módulos ofrece distintos servicios, entendiendo por servicio un flujo de datos de un tipo concreto, resultantes de sus distintas funcionalidades. Como resultado, se elimina cualquier tipo de jerarquía explícita, y la adición de un nuevo módulo se reduce a la resolución de nuevas relaciones de servicio, es decir, de tipado de datos.

Cada uno de los módulos, que corresponde a un proceso independiente desarrollado en C++, puede englobarse dentro de un nivel de abstracción distinto, de modo que cambiar el hardware sólo supondrá una modificación de los módulos de bajo nivel, haciendo muy portable tanto la arquitectura como las funcionalidades que implementa. Para la comunicación entre módulos, se ha hecho uso de la librería multiplataforma y de código abierto YARP [10], de la que se ofrece una descripción más completa en el Capítulo 3 de este texto.

Adelantaremos aquí que esta librería ofrece una abstracción de las comunicaciones entre procesos basada en el concepto de puerto. Cada puerto puede conectarse con otros para establecer un flujo de datos, cuyo tipado puede venir definido fácilmente por el usuario programador de C++.

Concretamente, en el proyecto URUS se ha diseñado una jerarquía de clases para los distintos módulos, partiendo de una clase base de tipo comunicaciones, cuya declaración recogemos a continuación.

```
class Comms: public yarp::os::Portable {

public:
        //! Friends:
        friend std::ostream& operator << (std::ostream& os,
            Comms& comms);
        friend std::istream& operator >> (std::istream& is,
            Comms& comms);
        friend std::ostream& operator << (std::ostream& os,
            Comms* comms);
        friend std::istream& operator >> (std::istream& is,
            Comms* comms);

        //! Data...
        double m_sec;   // Time from 1st january 1970 [s]
        double m_nsec;  // To increase time resolution [ns]
        int status;     // Status byte (system, error, ...)

        // Default constructor
        Comms(): m_sec(0.0), m_nsec(0.0), status(0) {}

        static void connect(std::string writer, std::string
            reader,  const char* defaultLocal = "tcp");
        virtual void print(PrintType); // Self-printing
           function (for debugging)
        virtual std::string logHeader();// Log-related
           function:

        // Timing functions:

        double tic();   // generates timestamp and keeps it
            in public data members
                        // m_sec, m_nsec; returns seconds
                           passed since 1970.
        double toc();   // returns seconds passed since
           last tic() function-call.

protected:

        // Log-related functions:
        virtual void logStream(std::ostream& os);
        virtual void fromStream(std::istream& is);

        // struct to get time marks
        struct timespec epochTime;
};
```

Tabla 2.1: ***Proyecto URUS – Comms.h.***

En la figura 2.10 puede verse un esquema de conexión entre los distintos módulos correspondientes al robot Romeo-4R.

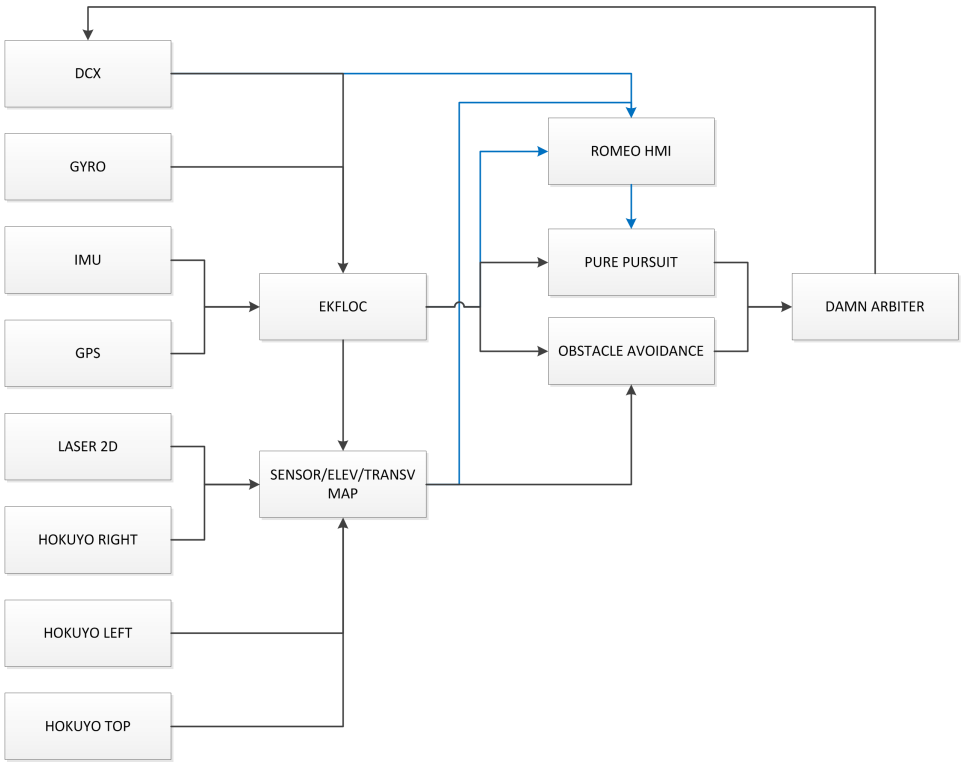

Figura 2.10: Conexiones módulos software Romeo-4R y puertos YARP.

En este esquema no se debe confundir cliente con *receptor de información* o servidor con *emisor de información*. En este sentido, un módulo llamado árbitro, que como se verá es el único que manda consignas al módulo de control de bajo nivel, ofrece un servicio que se podría definir como *consideraré los votos que reciba para la elección del comando que ejecutará finalmente el robot*. Para ello, los clientes deberán establecer la conexión, entendiéndose así que solicitan el servicio de este módulo. En este ejemplo queda claro que el servidor puede perfectamente recibir información de los clientes y prestar un servicio al mismo tiempo.

Otro ejemplo a considerar es el módulo de generación de mapas, que ofrece como servicio el mapa sensorial más actual y a su vez es cliente de los módulos de láser y del módulo de localización. Por otro lado, el módulo de evitación de obstáculos es cliente del generador de mapas, del módulo de localización y del árbitro; el servicio que ofrece se definirá como *intentará llegar al objetivo que reciba, evitando al mismo tiempo los obstáculos que haya en el camino*. Por ultimo, el módulo de tracking láser, que ofrece como servicio una lista de objetos percibidos, con sus posiciones, velocidades y firmas, podrá ser cliente tanto de módulos

láser como del generador de mapas, a la vez que podrá pasarle puntos de destino al módulo de evitación de obstáculos, resultando entonces una persecución.

La complejidad subyacente al tratamiento de imágenes tanto en la representación de mapas como en el tracking láser, se ha visto aliviada en parte gracias a la utilización de funciones incluidas dentro de otra librería de libre distribución, OpenCV (Open Source Computer Vision) [6]. Se trata de una librería de funciones programables en C/C++ dirigidas y optimizadas principalmente para la visión artificial en tiempo real, y que es independiente del hardware, sistema operativo o gestor de ventanas utilizado. En el capítulo 3 se hace una descripción más completa de esta librería.

A continuación se realizará una descripción más detallada de cada uno de los módulos actualmente implementados en Romeo. Mientras que los módulos de bajo nivel se encargan principalmente de interaccionar con los sensores y actuadores específicos de cada robot, aquellos que desarrollan funciones de más alto nivel resultan más independientes del hardware, con lo que resultan totalmente portables a otros robots con la misma arquitectura, como es el caso de otro de los robots del laboratorio llamado Monster.

- **Módulo DCX**

  Módulo que se comunica con la tarjeta de control DCX, sirve la velocidad lineal y la curvatura función del ángulo que forman las ruedas delanteras, ambas variables medidas mediante encoders, además de recibir las referencias para los motores de tracción y de dirección, que se pasan al control de bajo nivel que implementa un par de PID's.

- **Módulo GYRO**

  Interfaz con el giróscopo, este módulo, como el resto de los relacionados con sensores, se limita a empaquetar cada nuevo dato y servirlo por un puerto YARP para que cualquier otro módulo (cliente) pueda tener acceso a él. En el caso del giróscopo, este dato se refiere al ángulo y velocidad de giro en torno al eje z de Romeo.

- **Módulo IMU**

  Módulo que lee a través del puerto serie los datos provenientes de la Unidad de Medida Inercial, empaqueta los datos relativos a los ángulos de Tait-Bryan (roll, pitch, yaw) y los sirve al resto de módulos que los necesiten.

- **Módulo GPS**

  Lee a través del puerto serie la trama GGA del dispositivo GPS y a partir de ella extrae y empaqueta la posición global del robot (latitud, longitud y altura o UTM), la desviación estándar de dicha posición y otras variables de

interés como el número de satélites disponibles junto una marca de tiempo local.

- **Módulos Laser 2D**

  Servidores de la información generada por el láser SICK-LMS220, o por los URG-04LX de HOKUYO, resultan una fuente de información que por sus características resulta especialmente adecuada para la detección de obstáculos y generación de mapas. Esta información se compone básicamente de los campos relativos al número total de medidas y distancias percibidas para cada ángulo del barrido, así como del rango máximo o desviación estándar de la medida. En todo caso, la interfaz resulta común a ambos dispositivos, ya que toda la información que se puede extraer del sensor se sirve dentro de la estructura de datos correspondiente.

- **Módulo EKFLOC**

  Módulo encargado de localizar el vehículo en un sistema de coordenadas global, ya sea definido a priori como en el caso de UTM, ya sea definido por la postura inicial del robot, para ello toma los datos curvatura y velocidad (odometría), del giróscopo y del GPS y los integra mediante un Filtro de Kalman Extendido (EKF), dando como resultado una estimación de la posición y orientación del vehículo.

- **Módulo SENSOR MAP**

  La necesidad de un mapa local que integrara la información sensorial, por ejemplo para la identificación de la situación actual dentro del paradigma situación-acción que emplea el módulo de evitación de obstáculos, fue la motivación para la creación de este nuevo módulo. Si bien puede no considerarse estrictamente necesario, ya que la información sensorial está disponible en los módulos correspondientes a los sensores, en su funcionalidad cumple misiones importantes de cara a la flexibilidad de la arquitectura y a la precisión de la información:

  - *Fusión sensorial:* la información relativa a obstáculos de todos los sensores se condensa aquí ya no es necesario ser cliente de cada uno de los sensores. Esto permite entre otras cosas añadir o eliminar sensores sin que ningún otro módulo tenga que tener conocimiento de ello, además de mejorar la fiabilidad mediante tratamiento estadístico de los datos.
  - *Memoria:* sin la cual sólo se podrá disponer de la información actual. Ajustable por parámetro, especialmente para la aplicación en la evitación de obstáculos es necesaria una solución de compromiso: no conviene mantener un recuerdo perfecto de todos los obstáculos, que no tienen porqué ser fijos.

En la figura 2.11 podemos ver un mapa generado por este módulo en un experimento realizado en los laboratorios de la Escuela Superior de Ingenieros de Sevilla.

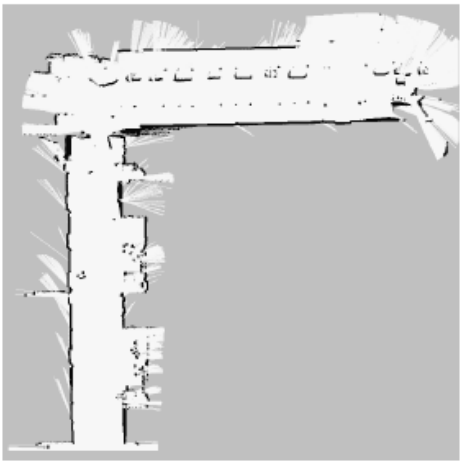

Figura 2.11: Mapa generado sensorialmente.

- **Módulo OBSTACLE AVOIDANCE**

  La funcionalidad de este módulo se basa en algoritmos clásicos de evitación de obstáculos para el cálculo de una acción que resulte en un movimiento dirigido a un objetivo y que al mismo tiempo evita obstáculos. Para ello se vale de una serie de diagramas, abstracción de toda la información relativa al problema. Inspirado por el trabajo de Minguez y Montano [21], no se trata de un planificador, sino de un algoritmo de navegación reactiva con ventajas respecto a la utilización de campos potenciales o incluso a la replanificación de trayectorias.

- **Módulo PURE PURSUIT**

  Módulo encargado de realizar un seguimiento estricto de trayectorias, tomará como entrada la estimación de la posición del robot para ejecutar el algoritmo de pure pursuit [14], un clásico en la navegación de robots móviles con direccionamiento Ackerman. Junto al módulo anterior, será cliente del módulo árbitro que veremos a continuación, que decide la acción definitiva a enviar al control de bajo nivel.

- **Módulo DAMN ARBITER**

  Este módulo estará encargado de arbitrar un numero variable de referencias en velocidad y curvatura, provenientes de diferentes módulos, de manera que las referencias finales que se envían al control de bajo nivel sean una solución de compromiso. Cada entrada tiene además un peso que indica al módulo su importancia relativa con respecto a las demás. De esta forma, la evitación

de un obstáculo próximo se antepondrá al seguimiento de una trayectoria previamente establecida, mientras que en ausencia de obstáculos se seguirá fielmente la trayectoria, tal y como se expone en [25].

- **Módulo LASER TRACKER**

  Diseñado para el tracking de objetos basado en percepción láser, parte de [15] para desarrollar un algoritmo de predicción y matching de firmas, formas percibidas, para en cada instante conocer la posición e identidad de cada objeto dentro del rango del sensor.

  Aunque se podrá usar también algoritmos de visión sobre mapas generados sensorialmente para la segmentación y extracción de los objetos, los retrasos variables hacen más adecuada en esta implementación concreta la utilización directa de los datos de sensores láser.

- **Módulo ELEVATION MAP**

  Módulo que integrando la información procedente del láser instalado en el techo del vehículo permite obtener en tiempo real un mapa de elevación del entorno y derivarlo en un mapa de transversabilidad del mismo, tal y como reflejan las imágenes de la figura 2.12.

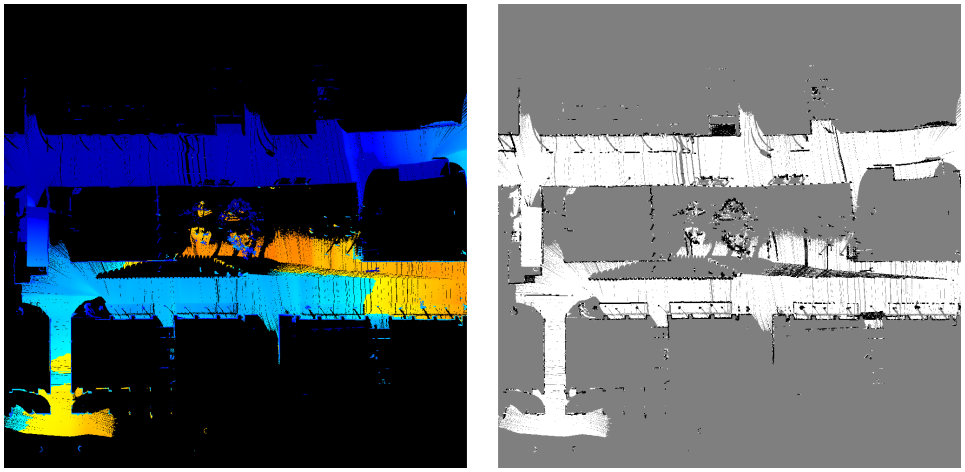

Figura 2.12: Proyecto URUS. Mapas de Elevación y Transversabilidad.

- **Módulo PORT LOGGER**

  Para la obtención de datos que permitan el posterior análisis y reproducción de los experimentos, se dispone también de este módulo, que se encarga de monitorizar y archivar en disco todo el flujo de datos que corre por la red YARP del robot.

- **Módulo LOG PLAYER**

Módulo que se utiliza para reproducir en simulación experimentos realizados previamente y cuyos datos obtenidos fueron logueados por el módulo PORT LOGGER.

Por tanto, no debe entenderse este módulo como un módulo más del sistema del robot. Se debe entender como un módulo o herramienta de apoyo al trabajo y al desarrollo del sistema completo, pues permite probar o refinar algoritmos de navegación o de filtrado de datos de sensores, sin tener que realizar un nuevo experimento real, algo que suele llevar mucho tiempo debido a la logística requerida para llevar a Romeo a un lugar donde poder realizar las pruebas reales.

El capítulo 5 de la presente memoria se dedica por completo a esta aplicación.

- **Módulo ROMEO HMI (Human Machine Interface)**

  La interfaz gráfica para el control y supervisión del funcionamiento del robot, objetivo principal del presente proyecto, se integra como un módulo más del mismo sistema, leyendo y escribiendo en diversos puertos YARP, permitiendo interactuar así con el robot. Esta aplicación fue diseñada pensando en mostrar información útil a los desarrolladores en tiempo real durante la realización de los distintos experimentos con el robot y no es en ningún caso una interfaz destinada a un posible usuario final.

Figura 2.13: Splashcreen Interfaz Romeo HMI.

El capítulo 4 de la presente memoria se dedica por completo a esta aplicación.