RoboCupJunior Rescue New Simulation 2021

Team Description Paper

Titán

## Abstract

1. Nuestro robot es capaz de generar una representación del mapa de la competencia y encontrar una secuencia de movimientos óptima para su navegación. Nuestro código tiene una arquitectura organizada y hemos desarrollado un conjunto de herramientas para facilitar la programación a alto nivel. Todo nuestro trabajo está documentado en github.

## Introduction

## Team

* + 1. Nuestro equipo está formado por Máximo Cansino, Lucas Flores y Alejandro de Ugarriza. Alejandro participó con otro equipo en la Primera Demonstración Simulada, Máximo y Alejandro fueron ganadores junto con otro compañero de la Roboliga Simulada en Argentina realizada en la plataforma de Erebus y Lucas es una nueva adición al equipo. Máximo se encarga de la detección de víctimas, Alejandro de mapeo y navegación y Lucas de la grilla entregable para finalizar el juego.

## Aim, strategy, and overall plan

* + 1. Nuestro objetivo es hacer un robot capás de navegar inteligentemente, analizar y mapear cualquier tipo de entorno. Calcular trayectorias optimas empleando complejos algoritmos para una navegación eficiente. Detectar víctimas y carteles empleando estrategias simples procesamiento de imágenes. Hacer un código más organizado, responsable, robusto y documentado a través de GitHub.
    2. Nuestra estrategia como equipo consiste principalmente en la división y asignación de roles para cada uno de los integrantes, coordinándonos a través de GithHub y reuniones virtuales una vez a la semana. Realizar una nueva arquitectura para el código y luego trasladar y modificar pedazos de nuestro código anterior allí, además de crear nuevas soluciones más robustas y eficientes.
    3. Empezamos realizando un esquema para la nueva arquitectura, definimos las clases en el código, luego empezamos a trasladar pedazos del código anterior, optimizándolos y adaptándolos a los cambios requeridos por la nueva plataforma. Continuamos definiendo el funcionamiento de sensores y actuadores básicos, luego los movimientos de precisión en el mapa, implementamos el mapeo del laberinto, luego los algoritmos de pathfinding y luego integramos todo y definimos la lógica de navegación. Paralelamente desarrollamos un sistema de detección de víctimas y carteles.

## Technical progress

## Navigation

* + 1. Nuestro Robot navega moviéndose en líneas rectas de vértice en vértice, un vértice siendo considerado a la escala de medias baldosas. Es capás de realizar secuencias de movimientos pre-programadas en situaciones clave.

Nuestro sistema de navegación está integrado por dos secciones: Análisis y Movimientos de bajo nivel.

* Análisis se encarga de planear la navegación a largo plazo para recorrer más efectivamente el mapa, y no tiene ninguna conexión directa con los actuadores. Utiliza entre otras cosas, breadth first search para encontrar la mejor casilla posible para moverse en la representación virtual del mapa. Una vez encontrada utiliza A Star para encontrar el camino mas corto para llegar a esta posición. Está encapsulado en la clase Analysis.
* La parte encargada de movimientos de bajo nivel tiene como función controlar los actuadores directamente para seguir el camino proveniente de Análisis o para sacar al robot de una emergencia que requiera control inmediato y directo. Esta distribuido entre la clase de RobotLayer y AbstractionLayer.

Ahora mismo tenemos únicamente los movimientos a bajo nivel en el nuevo código que desarrollamos para esta competencia, aunque tenemos gran parte de Análisis hecha en códigos anteriores, pero sin implementar.

* + - * Dividimos la navegación en corto y largo plazo.
      * Utilizamos el gps para obtener la orientación del robot por medio de diferencias entre posiciones. Esto nos permite definir un punto de referencia y mantenerlo ya que, al teletransportar al robot de una posición a otra, el giróscopo no es capaz de registrar estos cambios de orientación.
    1. Lo que nos falta es hacer que el robot se mueva de acuerdo a la planificación de movimiento a largo plazo. En un futuro planeamos implementar un sistema más complejo para encontrar las casillas óptimas para movernos y optimizar el movimiento a corto plazo para lograr más rapidez y fluidez. A muy largo plazo quizás incluyamos herramientas de Inteligencia Artificial más complejas e incluso Machine Learning.

## Detection

## Nuestro programa contiene un sistema de reconocimiento basado en distintos estados controlados por el panel de cámaras el cual analiza y detecta distintos tipos de objetos basado en su color, forma y área. Estados:

## Estado de escuchando

## Estado de clasificación de colores

## Estado de clasificación de formas

## En el estado de oyente la imagen de la cámara se divide en tres filtros principales: “Red panel”, “White panel” y “Yellow panel”. Cada uno de estos filtros está configurado para tomar un color especifico en un rango según matiz y saturación. Dichos valores fueron tomados a partir de múltiples pruebas en distintos entornos con la intención de que solo funcionen en determinados objetivos con el menor ruido posible. Es capaz de enviar el ángulo y el tamaño en píxeles de las víctimas o Hazard maps que detecte. Una vez que la cámara esté cerca de la víctima se calibra la perspectiva de la imagen y se clasifica de acuerdo a su forma, área y color.

## La forma de identificar o clasificar las víctimas no se basa en un algoritmo complejo, sino simplemente aplicando filtros lo que lo hace sencillo de implementar.

## En el futuro tenemos planeado implementar un sistema de comparación de matrices y detección de patrones a procesar para optimizar el programa en cuanto a tiempo y costo de computo.

## Mapping

* + 1. Para mapear utilizamos el sensor LIDAR, la cámara y el sensor de color. Los datos de estos son procesados y almacenados en una grilla. Para procesar los datos del LIDAR se comparan con distintas “plantillas” (formas de paredes, obstáculos, etc.). Por medio de las cámaras identificamos los tipos de víctimas y hazard maps. Con el sensor de color identificamos distintos tipos de suelos. En la grilla están representados cada baldosa (tomando como referencia las baldosas chicas), vértice y pared como objetos con diferentes propiedades. La grilla tiene la capacidad de ampliarse dinámicamente a medida que crece el mapa. Todavía no funciona la detección de obstáculos y paredes curvas, y casillas con el sensor de color.
    2. Utilizamos un sistema de plantillas para procesar los datos del LIDAR. Para evitar el ruido en los datos del LIDAR se implementó una cola de nubes de puntos para filtrar datos aleatorios inconsistentes.
    3. Para la detección de obstáculos y paredes curvas podemos utilizar el sistema de plantillas ya implementado. En cuanto a la detección de casillas de distintos tipos no tenemos resuelto un sistema para mapear la casilla entera, solo de a medias baldosas.

## Hardware

## Nuestra elección de los sensores se basa en lo que nosotros pensamos que era lo mas conveniente para poder sortear todos los problemas del laberinto de una forma más optima. Nuestro robot cuenta con los siguientes sensores:

## Lidar: elegimos este sensor en cuenta de los sensores de distancia debido a que nosotros pensamos que este sería más efectivo que los dichos censores ya que tenemos ventajas con la detección de paredes curvas, más distancia de lectura y más ángulos de medición. También elegimos este sensor ya que consideramos que sería algo interesante probar algo nuevo con lo que ninguno de nosotros habíamos trabajado antes. Este está ubicado arriba en el centro de forma que podamos aprovechar al máximo los 360° de medición del censor.

## Dos cámaras: Estas cámaras fueron elegidas con el objetivo de poder detectar e identificar los carteles y víctimas. Principalmente necesitábamos tres cámaras, pero debido al presupuesto solo pudimos comprar dos, ante este problema las posicionamos de forma estratégica a los laterales del robot para no perder ángulo de visión y compensar la cámara faltante.

## Giróscopo: Utilizamos el mismo con el objetivo de tener un buen seguimiento de la orientación en la que el robot se está moviendo al momento de recorrer el laberinto, este se encuentra ubicado en el centro de forma que tengamos una medición precisa de la rotación del mismo.

## GPS: este es se implementa para poder obtener con el mismo las coordenadas de donde se encuentra el robot y poder orientarnos ante reubicaciones del mismo. Se encuentra en el centro del robot para minorizar el ruido.

## Sensor de color: el mismo es utilizado para detectar los distintos colores de las baldosas para poder saber si cambiamos de zona o si nos encontramos con algún obstáculo en frente del robot (por ejemplo: pantanos, posos, etc.). Lo ubicamos al frente en la parte inferior de forma tal que el mismo apunte al suelo de forma directa para poder tener una buena lectura del mismo

## Program flow and sequential code execution:

* + 1. Desarrollamos un sistema para poder controlar en Flow de nuestro programa en forma de una máquina de estados, además de una manera de insertar pedazos de código secuencial al programa, donde se pueden usar, entre otras cosas, delays sin afectar otras partes del código que deben ejecutarse continuamente. Esto nos permite hacer pruebas y secuencias predeterminadas muy fácilmente.

## Performance evaluation

* 1. Evaluate your performance of your robot.

## Conclusion

* 1. Como conclusión podemos decir que mejoramos nuestro programa y trabajo en equipo teniendo en cuenta la experiencia de la competencia pasada y aunque todavía nos quede mucho por hacer, vamos por buen camino y tenemos una buena trayectoria hacia el futuro.

## Appendix

N.B.: The appendix is NOT to continue writing the main text. It should be reserved for additional info if the reader is interested or curious to know more. Teams may link a link to external documentation as an alternative to the appendix.

## Reference