Escuela
Politécnica
Superior

Sistema de seguimiento de humanos para robots cuadrúpedos

Grado en Ingeniería Robótica

Trabajo Fin de Grado

Autor:
Jose García Jávega
Tutor/es:
Miguel Ángel Cazorla Quevedo
Francisco Gómez Donoso

Julio 2022

Universitat d'Alacant
Universidad de Alicante

Sistema de seguimiento de humanos para robots cuadrúpedos

Autor
Jose García Jávega

Tutor/es
Miguel Ángel Cazorla Quevedo
Ciencia de la computación e Inteligencia Artificial
Francisco Gómez Donoso
Ciencia de la computación e Inteligencia Artificial

Grado en Ingeniería Robótica

Escuela
Politécnica
Superior

Universitat d'Alacant
Universidad de Alicante

ALICANTE, Julio 2022

Preámbulo

La principal razón que me ha motivado a decidirme por el desarrollo de este proyecto es el interés por este tipo de robótica móvil, la cual apuesta por la innovación al dejar a un lado los sistemas de locomoción tradicionales como serían las ruedas e innovar buscando sistemas más versátiles y a la vez más complejos como puede ser la aparición de extremidades para la locomoción y tratar así de expandir los conocimientos más allá de las simples plataformas móviles a las que estamos acostumbrados.

Agradecimientos

Agradecer a mis compañeros de clase, con quienes he compartido innumerables proyectos a lo largo de la carrera y si he llegado hasta aquí con estos conocimientos ha sido gracias a su ayuda y su apoyo.

A Francisco Gómez y a Miguel Ángel Cazorla por haberme acompañado y guiado a lo largo de este proyecto, ya que sin ellos nada de esto habría sido posible.

A mis padres y mis hermanas por todo el apoyo
a lo largo de esta etapa universitaria

Las especies que sobreviven no son las más fuertes,
sino aquellas que se adaptan mejor al cambio

Charles Darwin.

Índice general

1 Introducción **1**

2 Marco Teórico **3**
2.1 Algoritmos de detección de personas con cámara 4
2.1.1 Histograma de Gradientes Orientados 4
2.1.2 Mean Shift 4
2.2 Ejemplos de los primeros intentos de detección de personas empleando cámaras 5
2.3 Algoritmos de detección de personas con Lidar2D 5

3 Objetivos **7**

4 Metodología **9**
4.1 Ros 9
4.2 Python 9
4.3 Numpy 10
4.4 Gazebo 10
4.5 Estudios de la cinemática del robot 10
4.6 Darknet ros 10
4.7 Algoritmo de detección de Yolo 11
4.8 Actores 11
4.9 Paquete Chvmp 11
4.10 Repositorio de Unitree Robotics 11
4.11 Láser Lidar2D 12
4.12 Cámara ASUS simple 12
4.13 Aliengo 13

5 Desarrollo **15**
5.1 Simulación de personas utilizando Gazebo 16
5.2 Simulación de un robot cuadrúpedo utilizando Gazebo 17
5.3 Teleoperación del robot en Gazebo 17
5.4 Métodos para el seguimiento y detección de personas 18
5.5 Seguimiento de personas con Lidar2D 19
5.6 Detección de personas con cámara 20

6 Resultados **23**
6.1 Simulación de un personas en Gazebo 23
6.2 Simulación y teleoperación de un robot cuadrúpedo utilizando Gazebo 23
6.2.1 Empleo de el Github propio de Unitree Robotics 23
6.2.2 Búsqueda de otro software 24

6.3 Seguimiento de personas con Lidar2D 26
6.4 Detección de personas con cámara 33
6.5 Mejoras empleando Lidar2D 37

7 Conclusiones **39**

8 Trabajos Futuros **41**

Bibliografía **43**

Lista de Acrónimos y Abreviaturas **45**

Índice de figuras

1.1 Esquema del proyecto 2

2.1 Imagen presentación Aliengo de Unitree 3

4.1 Esquema de funcionamiento de ROS 9
4.2 Esquema cinemática 10
4.3 Esquema de barrido del sensor Lidar2d 12
4.4 Ejemplo de cámara ASUS 13
4.5 Diferencia entre Aliengo real y simulado en Gazebo 13

5.1 Esquema procesos y resultados proyecto 15
5.2 Ejemplo de actor 16
5.3 Menú teleoperación del paquete Chvmp 18
5.4 Sectores del sensor Lidar2d del robot 19
5.5 Reconocimiento de personas con darknet ros 20
5.6 Menú darknet ros 21

6.1 Menu del asistente de chvmp 24
6.2 Entorno de Gazebo sin obstáculos 25
6.3 Entorno de Gazebo con obstáculos 26
6.4 Gráfica con ruido sensores lidar2d 27
6.5 Entorno de Gazebo con obstáculos 27
6.6 Detección del actor primer intento 28
6.7 Detección del actor primer intento 28
6.8 Solución para poder detectar personas empleando lidar2d (sin transparencia) 29
6.9 Gráfica con las medidas obtenidas en Lidar2d 29
6.10 Primer paso en simulación Lidar2d 31
6.11 Segundo paso en simulación Lidar2d 31
6.12 Tercer paso en simulación Lidar2d 32
6.13 Cuarto paso en simulación Lidar2d 32
6.14 Quinto paso en simulación Lidar2d 33
6.15 Ejemplo de imagen procesada con YOLO 34
6.16 Esquema división en secciones de la cámara 35
6.17 Resultado Yolo a larga distancia 36
6.18 Resultado Yolo a corta distancia 36
6.19 Problema al usar Yolo a una distancia demasiado corta 37
6.20 Representación ejemplo obtenida en Rviz con el detector SPAAM 38

1 Introducción

En la actualidad, el diseño de sistemas de locomoción de robots con extremidades consiste en llevar a cabo un estudio sobre los sistemas biológicos presentes a nuestro alrededor.

Tales sistemas se pueden simplificar como un sistema de locomoción basado en un conjunto de eslabones que se hayan unidos por unas articulaciones, formando así a lo que, hoy en día, denominamos "patas".

De este modo, los seres vivos son capaces de recorrer terrenos irregulares, empleando un apoyo discreto de las patas sobre el terreno.

Debido a ello, existen máquinas caminantes que aprovechan este tipo de locomoción cambiando el número de puntos de apoyo; existiendo así, entre otros, los robots monópodos, bípedos, cuadrúpedos, hexápodos, etc. Gracias a esta característica, estas máquinas se convierten en el perfecto acompañante para una persona en cualquier tipo de entorno.

Por otra parte, estas máquinas presentan una serie de dificultades al contar con sistemas mecánicos y de control más complejos que los usados por plataformas móviles con ruedas, así como mayores niveles de sensorización, por lo que los costos de fabricación y mantenimiento son mucho más elevados que otros tipos de robots. Debido a esto, la cantidad de simulaciones relacionadas con robots cuadrúpedos no es nada comparada con la simulación de otros robots con una complejidad menor debido al empleo de ruedas para su locomoción.

Todas estas razones han llevado a que la finalidad principal de este proyecto sea la simulación de un robot cuadrúpedo; de modo que, este robot cuadrúpedo sea capaz de seguir a una persona por sí mismo y sin la necesidad de que un operario tenga que estar teleoperándolo. En este caso, el modelo de robot escogido será el Unitree Aliengo,con el que se pretende conseguir al final del proyecto que sea capaz de detectar personas en cualquier tipo de entorno y, posteriormente, tratar de llegar a su ubicación, logrando un seguimiento que conlleve en todo momento el respeto de las distancias de seguridad. Teniendo como objetivo evitar cualquier tipo de peligro que pueda surgir tanto para la persona como para el robot.

Así, al final del proyecto, cualquier persona que desee crear su propia aplicación pueda hacerlo siguiendo este proyecto y mejorarlo, ya que como se ha mencionado, las aplicaciones de este tipo de robots son inmensas.Entre algunos de los ejemplos que se observan como en el caso de que sea el propio robot el que nos guíe a nosotros, pudiendo emplear el robot como un perro lazarillo o que nos acompañe a nuestro lado y no siguiéndonos simplemente, también se podrían apreciar otras futuras aplicaciones para el transporte de objetos pesados como ya se ha visto en otros robots cuadrúpedos, este sería el caso del spot.

Por tanto, la innovación que se plantea con este proyecto es la posibilidad de que cualquiera pueda realizar una aplicación innovadora y revolucionaria empleando un robot cuadrúpedo para ello, ya sea teleoperándolo o controlándolo de forma automática. Es por ello que, en todo momento, se facilitarán las herramientas empleadas; debido a que, actualmente no hay muchos recursos enfocados a la simulación de robots cuadrúpedos y gracias al paquete de Chvmp se puede adaptar fácilmente todo lo empleado a lo largo de este proyecto para cambiar el modelo

del robot cuadrúpedo por cualquier otro que se desee.

Con este proyecto, se pretende simular un robot cuadrúpedo, controlando este mediante teleoperación para finalmente controlarlo de forma autónoma empleando sensores como una cámara o un sensor Lidar2D, logrando realizar tareas como el seguimiento y detección de personas en entornos no estructurados, por lo que también se realizarán simulaciones de personas dentro del entorno de Gazebo, analizando como se comportan estos modelos en las simulaciones. El esquema del proyecto es el siguiente:

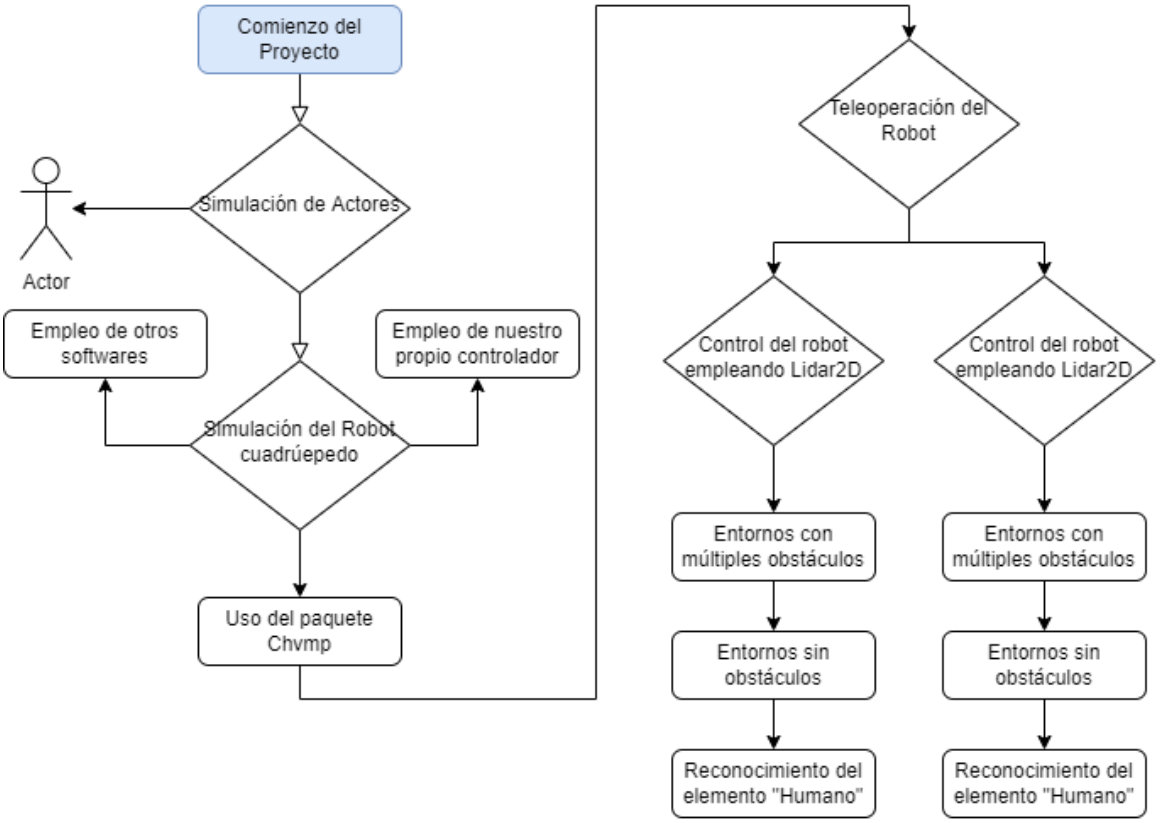

Figura 1.1: Esquema del proyecto

2 Marco Teórico

A día de hoy, no existen muchos proyectos que traten un problema como este, ya que los robots cuadrúpedos si que han sido empleados para la detección de personas utilizando el hardware propio. Sin embargo, a la hora de realizar una simulación el número de proyectos disminuye drásticamente, centrándose principalmente en la teleoperación y el mapeado con estos y no se centran en la detección y seguimiento de personas.

Algunos de estos proyectos son el proyecto de Chvmp para la teleoperación y mapeado empleando un robot cuadrúpedo, el cual ha sido la base del proyecto, ya que sin el uso de sus distintas órdenes de control no habría sido posible llegar a mover el robot, otros proyectos como el proyecto Wolf, creado por "Graiola" en el cual es similar al Chvmp, añadiendo recuperación a empujes, empleo de multirobots y control para subir escalones y demás obstáculos.

También se pueden destacar proyectos que logran una aplicación similar a la desarrollada en este proyecto; sin embargo, todos ellos empleando el hardware y los controladores reales, tal como muestra la propia empresa de Unitree Robotics en su vídeo de presentación.

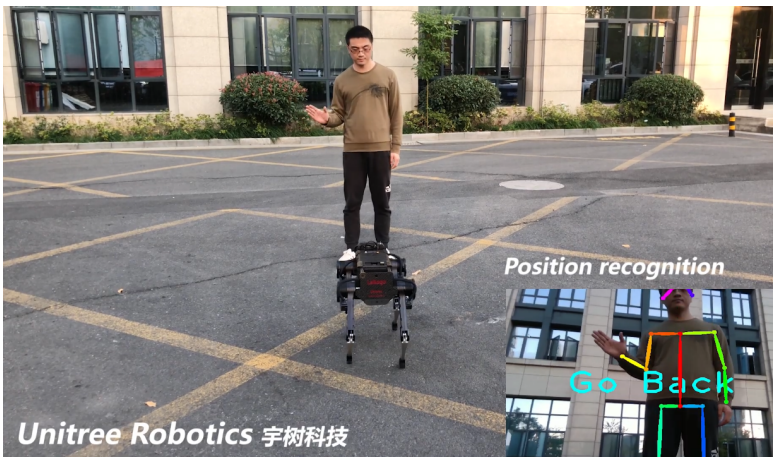

Figura 2.1: Imagen presentación Aliengo de Unitree

En esta presentación podemos ver la reacción del robot según la posición del cuerpo de la persona; sin embargo, no termina de realizar la tarea de seguimiento que se viene a tratar en este proyecto.

Por otra parte, hay varios proyectos que se centran en el seguimiento de personas y su detección en entornos aleatorios, que es el verdadero dato a tener en cuenta, pero estas simulaciones presentan la gran desventaja de que para ello emplean robots móviles con ruedas como podría ser un turtlebot o directamente no emplean robots, utilizando únicamente una cámara. Por lo tanto, no son tan versátiles ni complejos como un robot cuadrúpedo en este caso, puesto que un robot de estas especificaciones puede adaptarse mejor a los distintos

entornos que le rodeen y puede emplearse para múltiples tareas, las cuales si que pueden realizarse con un robot cuadrúpedo, como podría ser superar obstáculos tipo escalera o emplearse a modo de perro lazarillo o de compañía en tareas simples como caminar gracias a su forma similar a la de un perro.

Por tanto, la principal ventaja de este proyecto será dar a cualquiera la posibilidad de generar una aplicación ya sea teleoperada o automática sin la necesidad de tener el robot real, el cual suele ser necesario debido a la poca cantidad de proyectos enfocados en la simulación de este tipo de robots.

2.1 Algoritmos de detección de personas con cámara

En cuanto a las formas de detectar a una persona, actualmente hay numerosas formas, entre estas se observa normalmente un patrón definido por los siguientes pasos:

1. Adquirir imagen
2. Segmentar dicha imagen
3. Obtener las regiones de interés
4. Clasificar los objetos de interés

Entre estos métodos de detección se pueden destacar los siguientes:

2.1.1 Histograma de Gradientes Orientados

El método de reconocimiento de objetos mediante descriptores HOG, es un método que aún hoy sigue siendo ampliamente utilizado y con el que se obtienen buenos resultados. Esta técnica se basa en la cuenta de los gradientes de una zona de la imagen, defendiendo que un objeto puede ser perfectamente descrito por su distribución de sus gradientes de intensidad o dirección de su borde, por ello la imagen se divide en varias celdas y se crean histogramas con estas.

Este tipo de descriptor presenta algunas ventajas como es su invariabilidad a transformaciones, con la excepción de la rotación. Esta técnica resulta muy útil para el caso de detectar personas; puesto que, mientras se mantenga una posición erguida los movimientos corporales son ignorados.

2.1.2 Mean Shift

Se trata de un método muy eficaz, pero para ello es preciso conocer el entorno, debido a que presenta graves problemas si el fondo y el color del actor en nuestro caso son similares.

Otro de los problema que puede dar lugar a errores será la necesidad de tener a la persona y la cámara a la misma altura, lo cual no es posible en este caso tampoco. Por ello, no es recomendable utilizarlo para reconocimiento de personas, utilizando robots móviles como es el caso, ya que el entorno presenta un continuo cambio que, por lo general, será inesperado.

2.2 Ejemplos de los primeros intentos de detección de personas empleando cámaras

En 1998, Schlegel en el Instituto de Investigación para el Procesamiento del Conocimiento Aplicado, de la Universidad de Ulm, en Alemania, publicaron un trabajo donde presentaban un sistema para detectar y seguir a una persona con el objetivo de mejorar la interfaz persona-máquina de un robot autónomo móvil.

En el año 2000, Mckenna presentaron un trabajo titulado "Tracking Groups of People", el cual consiste en un sistema de visión por computador para el seguimiento de múltiples personas.

En el año 2005, B. Leibe en el articulo "Pedestrian detection in crowed scenes" se abordó el problema de la detección de peatones en las escenas de la vida real llena de superposiciones entre las personas.

En el año 2006, Dalal desarrollaron un detector para peatones, probado sobre secuencias de vídeos. Usando la técnica de histogramas orientados de flujo óptico diferencial combinado con los descriptores de apariencia de Histogramas Orientados a Gradientes, detectando el movimiento y segmentando el fondo.

2.3 Algoritmos de detección de personas con Lidar2D

En este caso, hay pocos métodos de detección de personas empleando sensores Lidar2D. Estos se basan principalmente en el movimiento; puesto que, si algo se mueve alrededor de ellos se supone que será una persona, sin embargo, esto acarrea una serie de problemas, uno de los problemas es la aparición de otros elementos moviles en nuestro entorno como puede tratarse de otro robot o animales, los cuales serían detectados también como personas, para evitar este problema se suelen emplear cámaras, con lo cual ya no estarían empleando este sensor únicamente. Un ejemplo de esta aplicación es el proyecto "A Method for Detecting Human by 2D-LiDAR" de Yuto Nagai o el proyecto denominado 2D lidar person detection del VisualComputingInstitute, cuyo repositorio de github es el siguiente: `https://github.com/VisualComputingInstitute/2D_lidar_person_detection`. No se han encontrado muchos algoritmos situados sobre una plataforma móvil empleando únicamente lidar2d. Entre los encontrados, se destaca en DROW detector y el Dr-spaam detector, presentando este último características bastante interesantes sobre la detección de personas en tiempo real.

3 Objetivos

Este TFG tiene como objetivo el estudio de los robots en este caso cuadrúpedos y la posterior simulación de uno, logrando detectar con este a una persona empleando dos tipos de sensores distintos, que en este caso se tratarán de un Lidar2D y una cámara. Por tanto se pueden definir los siguientes objetivos en este orden:

1. Simulación de un actor o humano en Gazebo
2. Simulación del robot en Gazebo
3. Teleoperación del robot en Gazebo
4. Empleo de un sensor Lidar2d para detectar y seguir al humano
5. Empleo de una cámara para detectar y seguir al actor empleando un sistema de detección

4 Metodología

En este apartado se va a hablar sobre las distintas herramientas empleadas para el correcto funcionamiento del proyecto:

4.1 Ros

Ros se trata del sistema operativo que emplearemos, este se trata de gran sistema operativo de código abierto, el cual facilita la comunicación entre los elementos que estemos simulando topics, a los cuales nos comunicaremos para publicar o recibir mensajes mediante el empleo de nodos suscriptores y publicadores, de este modo se puede obtener continuamente información de los topics para así poder conocer la situación de nuestro robot.

Un nodo vendría a ser un ejecutable que emplearemos para comunicarnos con los distintos topics, haciendo uso de funciones tipo callback en caso de ser un nodo suscrito a un topic o publishers en caso de estar publicando en un topic.

Un mensaje es el tipo de dato que se va a recibir o a enviar por los distintos topics.

Un topic es el canal por el que los mensajes serán tanto publicados como recibidos por los distintos nodos.

El nodo Máster es el nombre que recibe el servicio de ros.

Por tanto se establece el siguiente esquema con los distintos elementos mencionados:

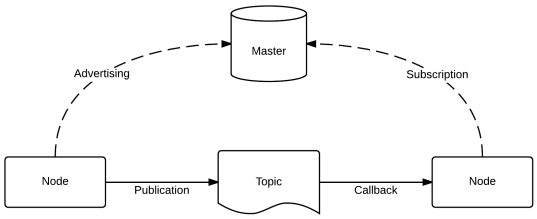

Figura 4.1: Esquema de funcionamiento de ROS

4.2 Python

Python es el lenguaje de programación que se utilizará principalmente durante el desarrollo del proyecto, este es un lenguaje de alto nivel de programación interpretado, en este caso será escogido, ya que ya es conocido y por su sencillez a la hora de crear los distintos callback para suscribirse y publicar en los distintos topics.

4.3 Numpy

Numpy es una librería de python, la cual está especializada en el cálculo numérico y el análisis de datos, para un gran volumen de datos, lo cual resulta ideal si queremos obtener información de topics y poder manejarla con precisión y sencillez.

4.4 Gazebo

Gazebo es un simulador 3D multi-robot con dinámica. Este nos ofrece la posibilidad de simular con precisión y eficiencia, diversidad de robots, objetos y sensores en ambientes complejos interiores y exteriores. Gazebo genera, tanto la realimentación realista de sensores, como las interacciones entre los objetos físicamente plausibles, incluida una simulación precisa de la física de cuerpo rígido.

4.5 Estudios de la cinemática del robot

Se emplearán algunas cinemáticas directas para obtener la posición de la pata y una cinemática inversa para obtener las posiciones articulares necesarias para alcanzar la posición deseada en el espacio, con esto se tratará de desarrollar nuestro propio controlador, empleando especialmente la cinemática inversa en este caso. La cadenas cinemáticas en robots cuadrúpedos se pueden describir de la siguiente manera:

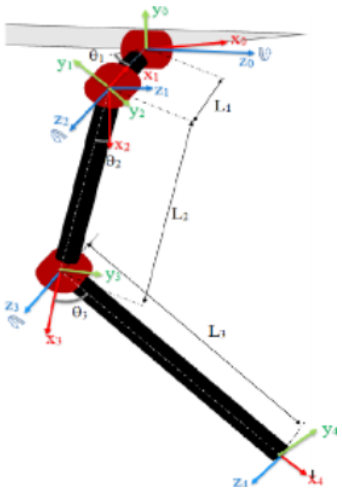

Figura 4.2: Esquema cinemática

4.6 Darknet ros

Darknet Ros es un marco de red neuronal de código abierto que se ejecuta en la CPU y GPU. Esta emplea el método de reconocimiento conocido como ”You only look once” (Yolo) el cual es un sistema de detección de objetos en tiempo real de última generación

4.7 Algoritmo de detección de Yolo

Dentro del paquete mencionado anteriormente, emplearemos esta red neuronal, la cual se caracteriza principalmente por su bajo coste computacional y por su gran precisión, lo cual lo hace ideal para tareas realizadas en tiempo real.

A diferencia del resto de métodos de detección este presenta un nuevo enfoque para la detección de objetos, ya que mientras que el resto se caracterizan por reutilizar clasificadores, YOLO enmarca la detección de objetos como un problema de una cuadrícula de celdas espacialmente separadas, prediciendo "Bounding Boxes" y probabilidades de clase en una única evaluación de la red neuronal. Como evalúa la imagen una única vez tanto en tiempo de entrenamiento como de ejecución, YOLO consigue una mayor velocidad una gran generalización de lo que representa un objeto perteneciente a una determinada clase.

Por tanto, su coste computacional derivado del empleo de una única evaluación de la imagen resulta ideal puesto que no solo reconoce que hay en la imagen, sino que también obtiene su posición, la cual destaca empleando las denominadas "Bounding Boxes", facilitando así la dirección en la que en un futuro querremos que nuestro robot se mueva.

4.8 Actores

Los denominados actores, que se tratan de modelos los cuales facilita Gazebo mediante un sencillo tutorial, con estos se pueden realizar trayectorias indicando la posición a la que se desea que vayan o se puede hacer uso de un plugin, el cual crea un actor que se mueve de forma completamente aleatoria.

4.9 Paquete Chvmp

Otro elemento a utilizar es el paquete de Chvmp, el cual trata de un paquete para el control y teleoperación de robots cuadrúpedos, entre sus funciones se pueden destacar también la del mapeado, en este caso, se emplea el paquete llamado Aliengo config y chvmp teleop con el fin de teleoperar y simular el movimiento del robot Aliengo de la empresa de Unitree Robotics. Más adelante se especificarán todos los sub-paquetes incluidos dentro de este y su funcionamiento, ya que con ellos se puede adaptar la aplicación a cualquier modelo de robot mientras que se disponga de un Urdf de este y algunas especificaciones como su altura, velocidad de los actuadores y ángulos limite de cada articulación

4.10 Repositorio de Unitree Robotics

Se tratan de los paquetes de simulación para ROS proporcionados para robots de Unitree. Con estos se pueden cargar robots y controladores de articulaciones en Gazebo, para que pueda realizar un control de bajo nivel de las articulaciones del robot. Teniendo en cuenta que la simulación de Gazebo no puede realizar un control de alto nivel, como es, caminar. Este repositorio emplea otros paquetes como es unitree-ros-to-real, con el cual se pueden controlar los robots reales en ROS. Para robots reales si que es posible realizar controles de alto y bajo nivel utilizando estos paquetes de ROS.

4.11 Láser Lidar2D

Para la primera detección de personas, se ha empleado un láser tipo Lidar2D, el cual presenta un rango de 260 grados en total realizando medidas cada 0.20 grados, para cada una de estas medidas se obtiene un valor con la distancia a la que se encuentra el primer elemento con el que el láser colisione, presenta un rango mínimo de 0.3 unidades y un rango máximo de 30 unidades en Gazebo, por lo que presenta potencia más que suficiente para la tarea a realizar.

Las medidas obtenidas en el rango se obtendrán de izquierda a derecha, siendo las medidas de la derecha las últimas dentro del vector que obtendremos por su correspondiente topic en Gazebo ("/scan").

Por tanto, tendremos una situación similar a la imagen 4.3

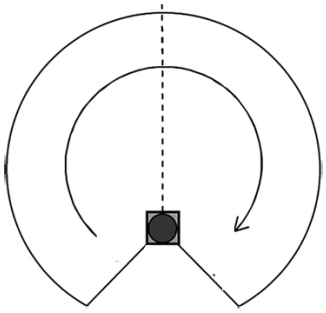

Figura 4.3: Esquema de barrido del sensor Lidar2d

Este dato se tendrá en cuenta más adelante, ya que será crítico a la hora de tomar las decisiones.

4.12 Cámara ASUS simple

Para la detección de personas, se ha empleado una cámara genérica de ASUS, la cual proporciona una matriz de dimensiones 640x640 con la que se obtiene una imagen de tipo RGB, dicha imagen será utilizada con el paquete darknet ros para la detección de personas.

Figura 4.4: Ejemplo de cámara ASUS

4.13 Aliengo

Aliengo es un robot de exploración, navegación y mapeo en tiempo real diseñado por Unitree Robotics. Este se caracteriza por ser ideal para entornos profesionales e industriales, también es una excelente herramienta para la investigación.

Este ofrece multitud funcionalidades de una plataforma robótica para aplicaciones autónomas de navegación, vigilancia y exploración, incluso en entornos hostiles.

Sin embargo, nuestro modelo no empleará los sensores tal como los emplea el robot real, por tanto se situará una cámara adherida en Gazebo bajo la cabeza del robot cuadrúpedo y un láser tipo Lidar2d sobre su espalda, esto no afectaría en un futuro a la aplicación en la realidad, puesto que solo sería necesario ajustar algunos parámetros como el rango del láser o la resolución de la cámara con el fin de no encontrarnos con errores no deseados.

El resto de partes del robot si que serán los propios del robot real, quedando la comparación entre el real y nuestra simulación del siguiente modo:

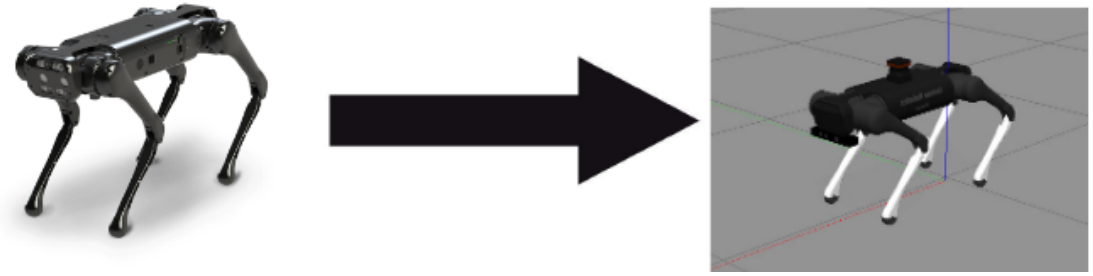

Figura 4.5: Diferencia entre Aliengo real y simulado en Gazebo

5 Desarrollo

En esta parte, se van a explicar las distintas partes de las que se compone el proyecto, así como la forma en las que estas se han abordado finalmente, el desarrollo se ha llevado a cabo de una forma lineal, simulando en un comienzo los actores, posteriormente el robot y finalmente se han creado los distintos sistemas de detección, los cuales se han desarrollado de forma conjunta. En el siguiente esquema, el cual se puede observar en la imagen 1.1 se reflejan todos los procesos por los que ha pasado el proyecto añadiendo un color distintivo según si se ha logrado o no obtener algún resultado útil de dicho proceso.

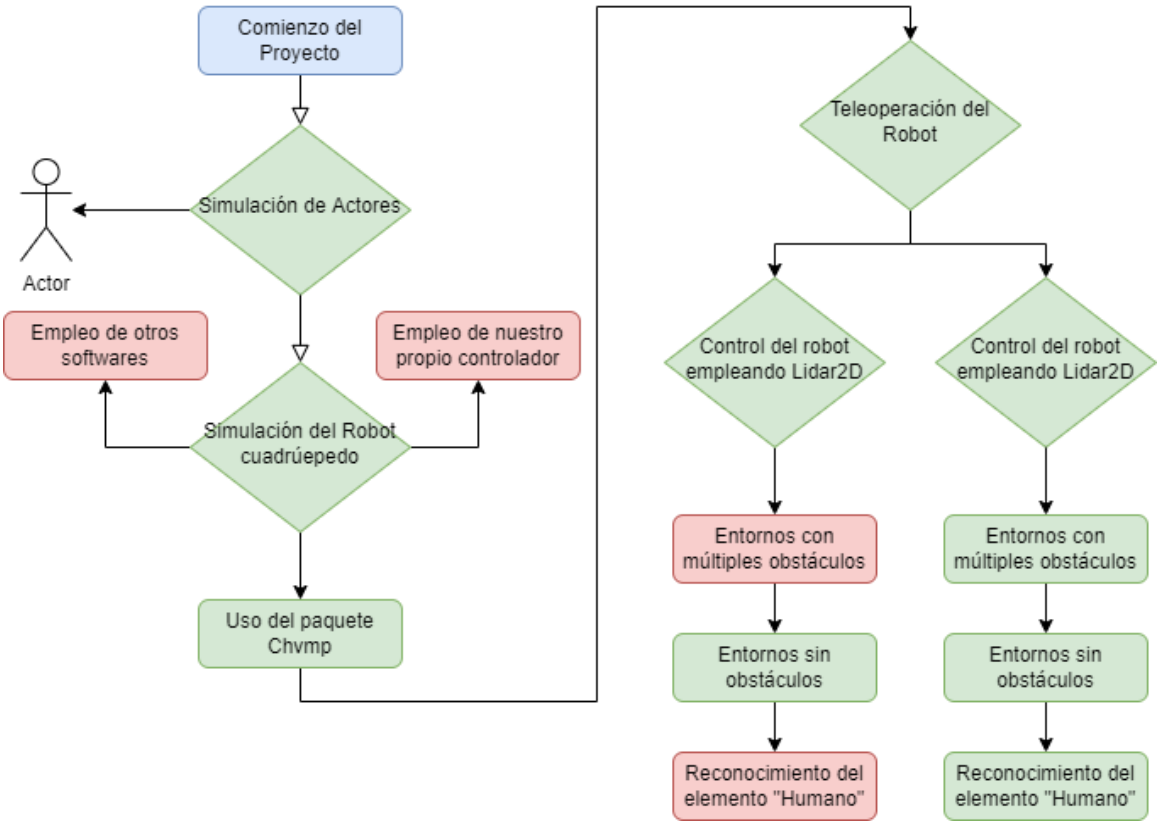

Figura 5.1: Esquema procesos y resultados proyecto

Como se puede apreciar, los resultados obtenidos con la cámara son mejores, esto es debido principalmente a que con la cámara obtenemos una cantidad de datos mayor como puede ser el color y una imagen completa de lo que el robot observa, además de la existencia de múltiples sistemas de detección más sofisticados para imágenes como es el caso de Yolo, el cual se emplea en este proyecto, mientras que con las lecturas del Lidar2D los datos obtenidos son

bastante inferiores en comparación y hay muy pocos métodos de detección, de los que se destaca DR-SPAAM.

5.1 Simulación de personas utilizando Gazebo

Para la simulación de personas en nuestro entorno, se utilizan los denominados ”actors”, de los cuales Gazebo nos facilita un modelo además de su respectivo tutorial para simularlos en cualquier mundo, para ello se introducirá el modelo del actor en nuestro file.world a simular y se definirá el tipo de animación que queremos emplear, empleando en este caso la animación ”walking”, una vez elegida las posiciones y tiempos por los que queremos que el ”actor” pase, definiendo así su trayectoria.

Con todo esto, se ha creado un mundo con un ”actor” donde se han observado algunas características sobre este, como que carece de colisiones cuando se encuentra con un obstáculo en la simulación.

Posteriormente, se ha encontrado un Plugin, en el cual se crea un ”actor” pero este se mueve de forma aleatoria, por lo que será útil durante las simulaciones con el robot para apreciar el funcionamiento del robot en situaciones más realistas, ya que nuestro robot deberá ser capaz de seguir a una persona sin conocer con precisión su trayectoria

Para la introducción de un actor en nuestro mundo será tan simple como crear un modelo tipo actor y posteriormente definir su trayectoria y tipo de animación en el .world como si se tratase de un objeto cualquiera. En el caso de querer introducir el plugin simplemente es necesario el código disponible en el tutorial de Gazebo, cambiando en este la posición donde queremos que aparezca por primera vez así como su animación.

Figura 5.2: Ejemplo de actor

5.2 Simulación de un robot cuadrúpedo utilizando Gazebo

Para la realización del proyecto, será necesario lograr una simulación en la que se puedan controlar los movimientos del robot cuadrúpedo sin que este se desplome o caiga debido a un mal control de sus extremidades, por ello se ha llevado a cabo una investigación de las distintas posibilidades para poder simular el robot, empleando como mejor opción el paquete de Chvmp para la simulación de cualquier tipo de robot cuadrúpedo, por lo que resulta ideal para poder simular el modelo deseado, el Unitree Aliengo.

Este paquete facilita la teleoperación de un modo sencillo, ya que traduce las órdenes de movimiento directamente a variables tipo "char"

Dentro de este paquete tenemos la opción de simular el robot tanto con Rviz como con Gazebo, pero primero se deberá crear nuestro propio paquete empleando el asistente de Chvmp, en el cual tendremos que especificar algunas características como la orientación de las patas, la altura a la que están situadas o sus velocidades máximas, una vez introducidos estos datos solo será necesario el Urdf con el modelo del robot de Unitree que cogeremos del Github de Unitree Robotics.

Para preparar el paquete con el asistente mencionado, es necesario acceder al github de chvmp setup assistant y clonar dicho repositorio en nuestro Workspace, no sin antes haber clonado también el repositorio del paquete chvmp, una vez clonado accederemos al asistente con el comando roslaunch champ-setup-assistant setup-assistant.launch Dentro de este simplemente será necesario escoger el URDF del robot a simular y especificar algunos parámetros.

Una vez realizados estos pasos se crea el paquete con todos los .launch necesarios para la visualización del robot, más concretamente el archivo bringup.launch, con el que se cargan los controladores del robot. Si se añade al final de este .launch la opción de rviz:=true en la terminal se obtendrá la representación en Rviz también como se ha comentado.

Por otro lado, si se emplea el archivo gazebo.launch, se cargan automáticamente los controladores y se inicia la simulación en Gazebo, obviamente sin que el robot haga nada más que mantenerse de pie puesto que no se le están enviando órdenes.

Destacar que es posible la aparición de un error si no se instalan los xacro necesarios para ejecutar la simulación, de ser así se deben instalar estos.

5.3 Teleoperación del robot en Gazebo

Una vez simulado correctamente el robot en el entorno, se puede emplear el paquete Chvmp teleop para controlar por teclado el movimiento del robot con las ordenes definidas en cada char, este presenta la opción de controlar el robot también por un joystick pero en este caso solo se usa el teclado, dentro de las posibilidades de movimiento, el robot se puede mover hacia delante, atrás o girar en ambas direcciones, pero no será posible cambiar la inclinación del mismo, también se presentan opciones para modificar la velocidad del mismo y aplicar movimientos omnidireccionales el menú con el que nos orientaremos para asignar a cada variable tipo char una orden es el propio de Chvmp 5.3

Para lograr esta teleoperación, simplemente es necesario lanzar la simulación en Gazebo mencionada en el apartado anterior y posteriormente lanzar el teleop.launch contenido en el paquete chvmp teleop, el cual ya se ha clonado antes, obteniendo así el menú especificado en la imagen 5.3, una vez aparece este menú simplemente se deben introducir por teclado las

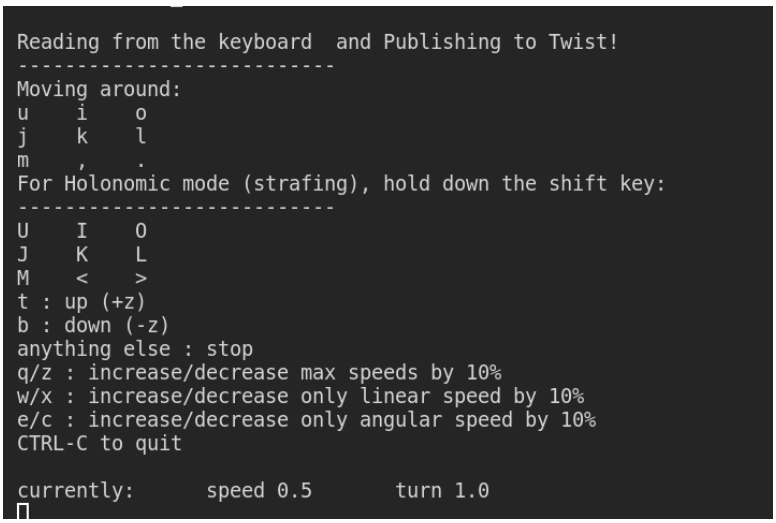

Figura 5.3: Menú teleoperación del paquete Chvmp

órdenes de movimiento y el robot se moverá en la simulación.

Es recomendable utilizar los movimientos de giro y avance y retroceso por separado, ya que utilizar ambos al mismo tiempo puede ocasionar la pérdida del equilibrio del robot.

Para cargar correctamente los controladores hay que tener en cuenta que el dominio de estos está anticuado, por ello se debe modificar manualmente mediante el comando por terminal "sudo gedit /.ignition/fuel/config.yaml" donde se debe sustituir el dominio https://api.ignitionfuel.org por el dominio https://api.ignitionrobotics.org

5.4 Métodos para el seguimiento y detección de personas

Una vez se ha controlado el robot mediante teclado, se tratará de hacer que este funcione de forma autónoma siguiendo a personas empleando la cámara o un Lidar2D tratando de seguir a los actores en tiempo real.

Debido a la ausencia de un sensor Lidar2D y de una cámara en el Urdf original de Unitree Robotics, se emplea el Urdf facilitado en el repositorio de Chvmp donde el robot presenta una cámara con imágenes RGB y un sensor Lidar2D sobre él, empleando estos dos tipos de sensores se ha desarrollado un software utilizando como base el software de la teleoperación para poder detectar y seguir a los actores creados anteriormente de forma consistente y evitando obstáculos.

Para ello, se emplea un script de Python, el cual es una versión modificada del archivo empleado para la teleoperación en el cual empleando constantemente funciones callback se obtiene información de los sensores y si es necesario del entorno de Gazebo para así procesar toda la información de estos y actuar en consecuencia, logrando así un comportamiento autónomo.

5.5 Seguimiento de personas con Lidar2D

Como ya se ha mencionado, se ha comenzado implementado una función para seguir a las personas cercanas empleando un Lidar2D, este sensor nos proporcionará un vector con todas las distancias de los elementos cercanos con un alcance máximo de 30 unidades en Gazebo, además de presentar un rango de un total de 260 grados, comprendiendo 130 grados hacia la izquierda y 130 grados hacia la derecha, de tal modo que se tiene un ángulo ”muerto” de un total de 100 grados en la parte posterior del robot.

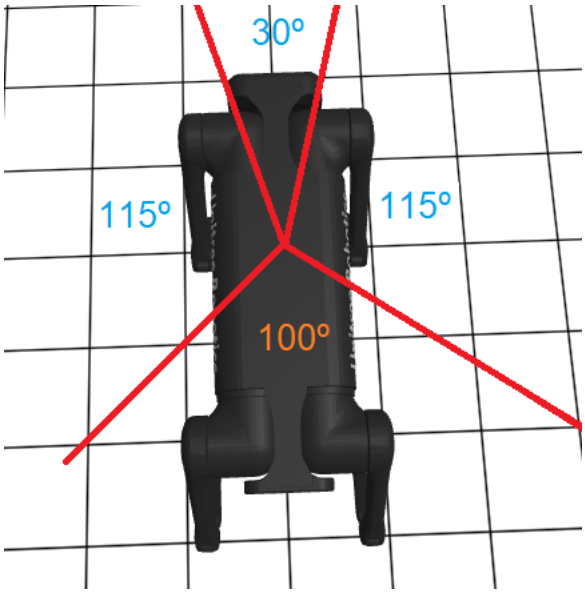

Figura 5.4: Sectores del sensor Lidar2d del robot

Estos datos serán recibidos mediante un topic, el robot obtiene una medida de distancia cada 0.25 grados, dicho valor es usado para lograr saber en qué dirección están los obstáculos.

La dificultad de este tipo de sensor consistirá en la diferenciación entre un objeto con las piernas humanas, ya que por parte del sensor solo se obtendrán medidas de distancia y no se conoce ningún otro tipo de dato que pueda resultar relevante para su diferenciación, por lo que se debe considerar la opción de presentar entornos completamente vacíos en el caso de que no se logre diferenciar entre los distintos elementos detectados.

Por ello, a día de hoy sigue siendo un gran desafío la detección de personas empleando un Lidar2d, debido a que este no proporciona normalmente información suficiente.

Por ello, se ha optado por emplear un entorno vacío y crear un nuevo fichero .launch que sustituya al empleado para la teleoperación, en este, nos suscribiremos a topics para obtener información del entorno para así poder introducir un objeto invisible en la posición del actor, garantizando así su detección.

Por tanto, la forma de ejecutar esta aplicación es la misma que la vista anteriormente, cambiando el el teleop.launch por el archivo que en este caso se ha llamado laser.launch

5.6 Detección de personas con cámara

Una vez creada la aplicación con el Lidar2D se ha desarrollado una aplicación empleando un algoritmo de detección asociado a la cámara, eligiendo el algoritmo conocido como YOLO (you only look once), el cual como bien indica su nombre es muy rápido haciéndolo ideal para la detección en tiempo real necesaria para controlar el robot como ya se ha explicado anteriormente. Por otra parte, este algoritmo presenta una fiabilidad muy alta además de proporcionar datos de la clase del objeto detectado (que en este caso el objeto es persona), además de proporcionar información sobre la ubicación del objeto mediante las coordenadas de este en la imagen y la probabilidad de que ese objeto se corresponda con el detectado.

Una vez conocido esto, se ha seleccionado el paquete de darknet ros, con el que aplicar la red YOLO a una imagen publicada en un topic, de este modo, se modificará el topic para que coincida con el de nuestra cámara, una vez detectada la imagen en el topic especificado con el paquete se mostrarán las imágenes formando un vídeo con la clasificación realizada y mostrando cuadros delimitadores alrededor de las distintas figuras que se encuentren tal como se ve en la imagen 5.5

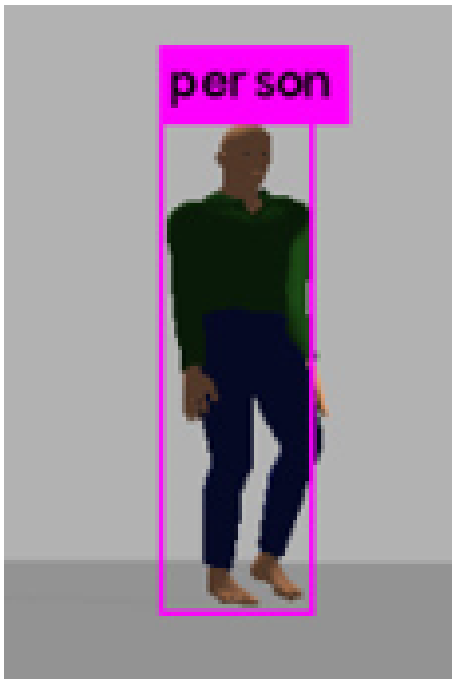

Figura 5.5: Reconocimiento de personas con darknet ros

Para el control del robot no se emplearán directamente estas imágenes, sino que emplearemos los distintos topics que este paquete crea, puesto que en ellos está la información realmente relevante como pueden ser las coordenadas en la imagen donde se sitúa la persona o el identificador con el que distinguiremos entre una persona o una silla, además de otro topic con el que podremos saber si la cámara está detectando algo, ya que cuando esta no detecta nada se dejan de publicar elementos en el topic empleado para obtener las clases de los objetos detectados.

Durante la ejecución del paquete, obtendremos por terminal y en tiempo real la probabilidad de que el elemento detectado pertenezca a una clase y los FPS de la detección en tiempo real como se ve en la figura 5.6

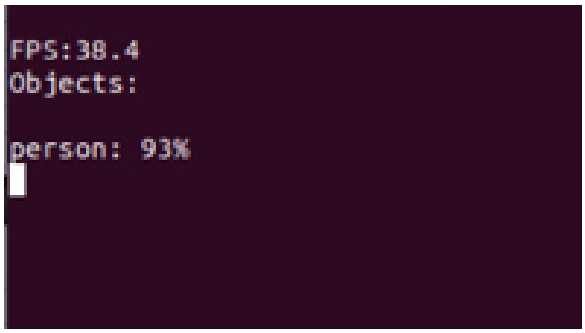

Figura 5.6: Menú darknet ros

Estos datos no serán empleados directamente desde aquí, ya que como se ha mencionado los leeremos directamente desde un topic aparte.

Para esta aplicación, se ha creado un .launch llamado cámara.launch que sustituye a láser.launch o a teleop.launch, sin embargo en este caso será necesario también lanzar el fichero darknet ros.launch dentro del paquete de darknet ros, este obtiene las imágenes publicadas en el topic que especifiquemos en la ruta "darknet ros/darknet ros/config/ros.yaml", una vez ejecutado este launch, se crearán los respectivos topics de darknet, con los que el robot comenzará a moverse.

6 Resultados

Tras explicar las distintas partes del proyecto y como se han planteado, en este apartado se reflejarán todas las alternativas realizadas hasta lograr los procedimientos empleados en el desarrollo.

6.1 Simulación de un personas en Gazebo

Antes de comenzar la experimentación con el robot como tal, se realizaron algunas pruebas con el fin de aprender a simular los actores de forma correcta, entre las pruebas realizadas se puede concluir que los actores presentan distintas animaciones para simular su movimiento, de las cuales se ha escogido la de walking, ya que resulta la más adecuada para la aplicación a realizar. Para lograr dichos movimientos simplemente se tendrá que especificar dentro de nuestro .world especificando la posición a la que se desea que el actor vaya mediante coordenadas XYZ, el tiempo que se emplea para ir de un punto a otro y la orientación con la que se desea que alcance la posición final.

6.2 Simulación y teleoperación de un robot cuadrúpedo utilizando Gazebo

Debido al tedioso problema que esta primera tarea provocó, se especificarán a continuación las etapas por las que se pasó hasta conseguir encontrar una forma de simular el robot de forma eficiente.

6.2.1 Empleo de el Github propio de Unitree Robotics

Como ya se ha especificado previamente, para simulación del robot cuadrúpedo se ha escogido el modelo Aliengo de la empresa de Unitree Robotics, por lo que se emplearon primero los paquetes para su simulación y control facilitados en el Github oficial de Unitree Robotics Dentro de este repositorio, llamado unitree ros, se emplean para la simulación del robot los paquetes unitree controller, unitree gazebo , unitree legged control y adicionalmente unitree ros to real donde se encuentra el paquete unitree legged msgs que también será necesario para la simulación, estos paquetes se acabarán usando para la simulación pero como paquetes secundarios asociados al paquete principal.

Una vez empleado el paquete de unitree gazebo, se consigue simular el robot, sin embargo no de la forma esperada, ya que dentro de estos paquetes se encuentra un controlador para levantar el robot, pero no para moverlo, por lo que se trató de emplear ese controlador creado en C++ para mover cada una de las patas a la posición deseada empleando cinemática inversa obteniendo las posiciones a las que se desea llegar y en las que estamos empleando los topics de gazebo, sin embargo, una vez realizada la cinemática inversa se acabó descartando la idea

de generar nuestro propio controlador, ya que el error acumulado sería excesivo, por lo que se pensó en utilizar otra función implementada en el paquete, la cual mueve el modelo generado publicando la nueva posición de cada una de las articulaciones, sin embargo este movimiento no resultaba muy natural al no moverse ninguna de las articulaciones y parecer que el robot se iba deslizando sobre el suelo, por lo que se descarta la idea de simular este robot empleando el paquete oficial de Unitree Robotics como paquete principal para la simulación, más adelante si que se empleará debido a que contiene el modelo del robot pero sin sus controladores.

6.2.2 Búsqueda de otro software

Una vez descartado el paquete de Unitree Robotics, se optó por buscar una forma de simular el robot y controlar su movimiento de cualquier forma, por lo que se optó por emplear el paquete chvmp diseñado para controlar robots cuadrúpedos, dentro de este paquete tendremos múltiples funciones como mapeado y teleoperación, en nuestro caso se emplea únicamente la teleoperación, por lo que no será necesario profundizar en el mapeo.

Dentro de este paquete tenemos la opción de simular el robot tanto con rviz como con Gazebo, pero primero se deberá crear nuestro propio paquete empleando el asistente de chvmp, en el cual tendremos que especificar algunas características como la orientación de las patas, la altura a la que están situadas o sus velocidades máximas, una vez introducidos estos datos solo será necesario el urdf con el modelo del robot de unitree que cogeremos del Github de Unitree Robotics. El paquete en cuestión se llama chvmp setup assistant, con el que se creará un paquete a partir del urdf y las especificaciones que se le indiquen.

Una vez lanzado el menú del asistente aparecerá la siguiente interfaz, donde se rellenarán los parámetros especificados:

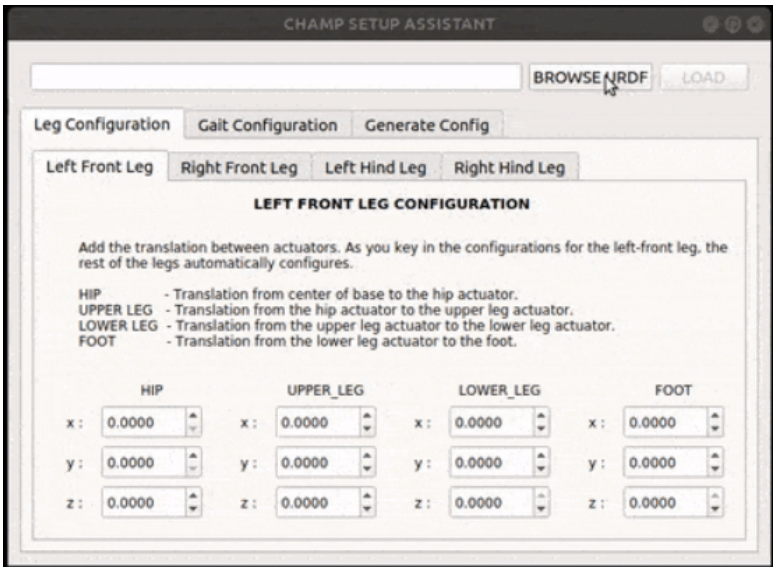

Figura 6.1: Menu del asistente de chvmp

Una vez rellenado los parámetros necesarios en el menú, se escoge un nombre para el paquete que el asistente crea, acompañado de los caracteres config, por tanto nuestro paquete

se llamará aliengo config

Ya con el paquete creado, se puede ver que en la simulación de rviz el robot no presenta ningún problema a la hora de teleoperarlo, sin embargo en la simulación de Gazebo el robot aparece tirado en el suelo y no es posible teleoperarlo.

Por ello, se volvió a crear el paquete usando el asistente modificando los distintos parámetros, sin embargo siempre aparecía el mismo problema o la simulación de rviz dejaba de funcionar correctamente, es por ello que se empleó el paquete facilitado en el mismo repositorio de chvmp para simular el Unitree Aliengo, pero aun así el problema persistía por lo que por el momento no se siguió empleando este paquete hasta un futuro, ya que el robot aparecía tirado en el suelo sin levantarse y sin seguir ninguna de las ordenes de la teleoperación.

Tras probar otras opciones como el paquete "Wolf" que resulta similar al anterior se decidió intentar solucionar de nuevo el problema encontrado con chvmp, ya que el problema reside en los controladores, los cuales no se cargan correctamente en la simulación de Gazebo debido a la activación de una función denominada como "REST" durante la ejecución del launch. Este error se debe a la activación de la función REST que se ha comentado, el cual se soluciona cambiando la url dentro del archivo "ignition/fuel/config.yaml", dentro de este se encuentra una Url la cual pertenece a un antiguo dominio, el cual no funciona correctamente, cambiando esta Url por "https://api.ignitionrobotics.org" se soluciona el problema y la teleoperación funciona perfectamente dentro de Gazebo, logrando así poder continuar con el proyecto de forma adecuada.

Una vez lograda la simulación, se han definido dos entornos en los cuales se llevan a cabo las pruebas, estos entornos se diferencian principalmente en la diferencia en el número de obstáculos, ya que el primer entorno contiene únicamente una persona además del robot.

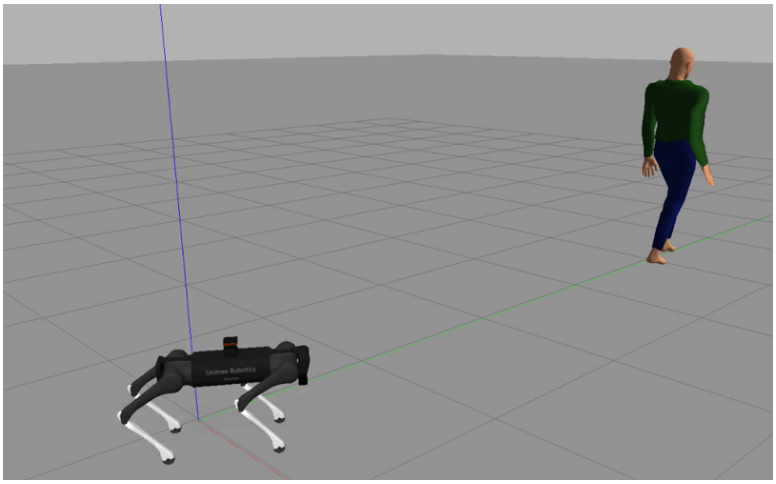

Figura 6.2: Entorno de Gazebo sin obstáculos

Mientras que por el otro lado, el segundo entorno presenta bastantes más obstáculos simulando una obra.

Una vez simulado correctamente el robot en el entorno con el actor, se han probado los distintos tipos de movimiento para observar si todos ellos funcionan correctamente, concluyendo que los movimientos de giro y avance/retroceso funcionan a la perfección, sin embargo,

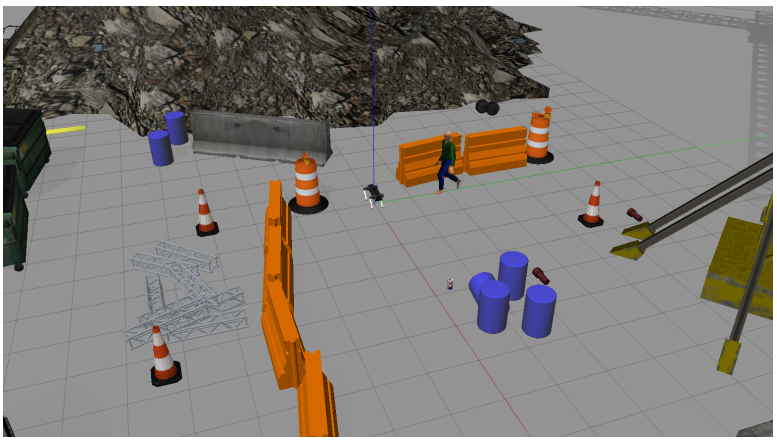

Figura 6.3: Entorno de Gazebo con obstáculos

los movimientos que unen avance con giro o retroceso con giro parecen funcionar, pero en ocasiones el robot pierde estabilidad dando lugar a caídas. Por otra parte, se ha probado a aumentar la velocidad del robot dentro del menú especificado en la figura 5.3, pero dicho aumento de velocidad parece aumentar drásticamente la inestabilidad de los movimientos, por lo que se dejará en su valor por defecto de 0.5.

Debido a la ausencia de un sensor lidar2D y de una cámara en el urdf original de unitree robotics, se emplea el urdf facilitado en el repositorio de chvmp donde el robot presenta una cámara con imágenes RGB y un sensor Lidar2D sobre él.

6.3 Seguimiento de personas con Lidar2D

Para comenzar a trabajar con el robot, se ha implementado una función para seguir a las personas cercanas empleando un lidar2D, este sensor nos proporcionará las medidas necesarias para realizarlo.

Estas medidas serán recibidas mediante el topic /scan, las cuales se transformarán las unidades angulares a grados para facilitar su compresión, de este modo también se observa que el robot obtiene una medida de distancia cada 0.25 grados, dicho valor es usado para lograr saber en qué dirección están los obstáculos.

Por tanto, en una primera instancia se ha desarrollado una gráfica con los datos obtenidos, cambiando las medidas en las que el robot no detectaba nada por 0, ya que de lo contrario, se obtiene un valor infinito, el cual no se puede representar correctamente en la gráfica, a esto añadir el empleo de un pequeño umbral, con el cual se corregirán algunas medidas erróneas obtenidas con el láser, ya que este obtiene medidas entre 0 y 0.39 unidades en zonas en las que no hay ningún elemento, por lo que se han suavizado y eliminado, haciendo así que el robot no detecte obstáculos a medidas inferiores a 0.4 unidades, de lo contrario se obtienen los siguientes resultados en un entorno con una esfera y el robot.

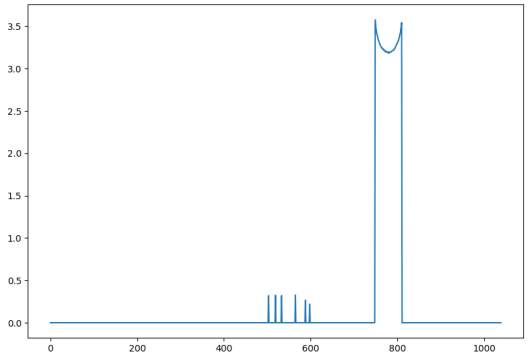

Figura 6.4: Gráfica con ruido sensores lidar2d

Donde se puede apreciar la aparición del ruido ya mencionado, el otro elemento se trata simplemente de la esfera que se ha simulado en un entorno vacío para obtener algunas lecturas.

Una vez solucionado el error del ruido, se ha comenzado a trabajar sobre el entorno con múltiples obstáculos asegurando la ausencia de ruido y errores por parte del sensor. Al ob-

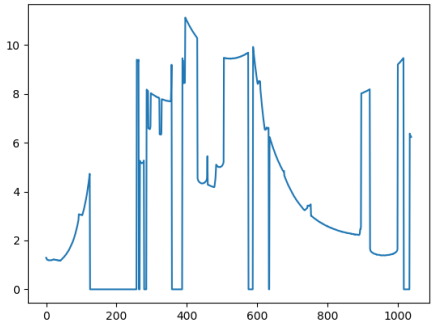

Figura 6.5: Entorno de Gazebo con obstáculos

servar esta gráfica del robot en el entorno con obstáculos surge un gran problema, no somos capaces de diferenciar cual de todos los obstáculos se corresponde con el del actor, por lo que se descartó en un principio la detección del actor mientras tuviésemos más obstáculos cerca.

Por tanto, se comenzará en el entorno descrito anteriormente, donde solo presentaremos un actor y el robot, facilitando así su reconocimiento.

Sin embargo, al tratar de detectar el ”actor” en nuestro entorno vacío, nos damos cuenta de que al carecer este de físicas no es posible obtener ninguna medida relacionada con él, por lo que esto dificulta gravemente su detección, por ello se trató de emplear una cámara tipo GPU con el fin de detectar el esqueleto de este actor, ya que es posible, sin embargo al cambiar dicha cámara la simulación fracasaba drásticamente, por lo que se descartó esa idea.

El error descrito se puede observar en la imagen 6.6, donde por algún motivo el robot aparece sobre el actor con las extremidades completamente rectas y el actor aparece tumbado sin moverse en ningún momento. Se trató de solucionar cambiando distintos parámetros

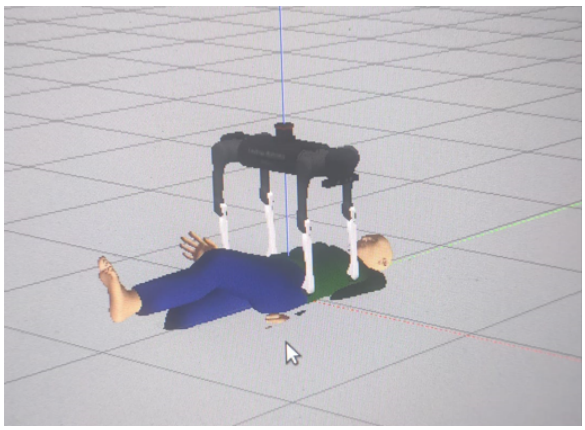

Figura 6.6: Detección del actor primer intento

dentro del Urdf del robot y de la cámara para detectar el problema pero no fue posible encontrar una solución, por lo que se prosiguió con la cámara normal.

Los resultados obtenidos en un entorno con el actor fueron el primer intento que se realizó con el fin de observar las medidas obtenidas al detectar a dicho actor.

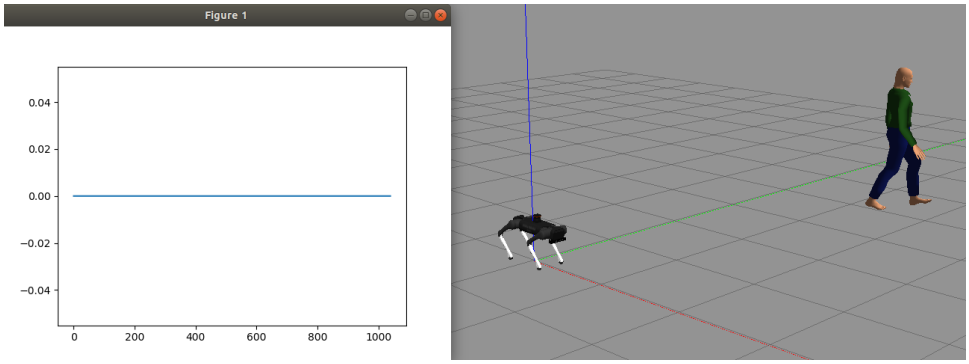

Figura 6.7: Detección del actor primer intento

Como se puede apreciar, en la figura 6.7 las medidas obtenidas por el robot son todas de 0, por lo que no detectamos absolutamente nada.

Finalmente, llegué a la conclusión de que una posible solución podría ser la creación de un elemento invisible dentro de nuestro entorno, el cual si que se pudiese detectar con el láser y que no afectase a la simulación, es por ello que se creó un cilindro el cual actualiza su posición para situarse siempre en la posición del actor, al no presentar colisiones este, ninguno de los 2 modelos (actor y cilindro) presentan ningún problema al estar uno dentro del otro, para lograr dicha actualización de la posición del cilindro se emplea una función callback la cual suscribiendose al topic /gazebo/model states obtiene de todos los modelos representados en el entorno la posición del modelo cuyo nombre sea ”actor”, posteriormente se publica por el topic /gazebo/set model states un mensaje con el nombre ”unit cylinder” con las posiciones

X e Y del actor, de este modo se logra mover el cilindro a dicha posición, no sin antes reducir el valor de la coordenada Z en una unidad, haciendo así que el cilindro quede cerca del suelo para así garantizar su detección con el robot, puesto que este no presenta una altura muy elevada en comparación con la de una persona. Si el elemento descrito fuese visible se vería del siguiente modo, obviamente durante las simulaciones se aumentará su transparencia al máximo para que no sea visible y los resultados obtenidos sean más realistas.

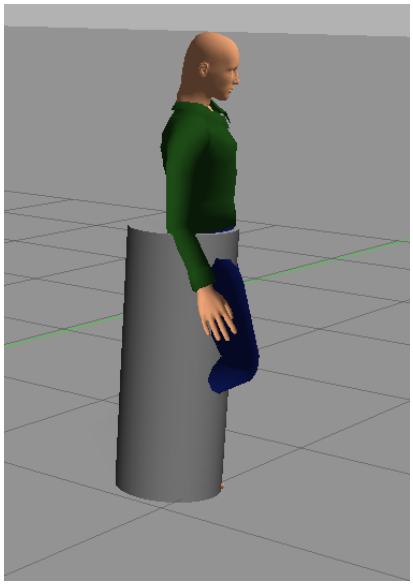

Figura 6.8: Solución para poder detectar personas empleando lidar2d (sin transparencia)

Por tanto, debido a este cambio, las lecturas obtenidas por el láser no serán exactamente las de dos piernas humanas, sino que se reflejará más la forma de un semicírculo, puesto que este cilindro carece de inclinación por lo que no se puede llegar a decir que se trate de una simulación de una pierna, sino más bien del tronco de la persona.

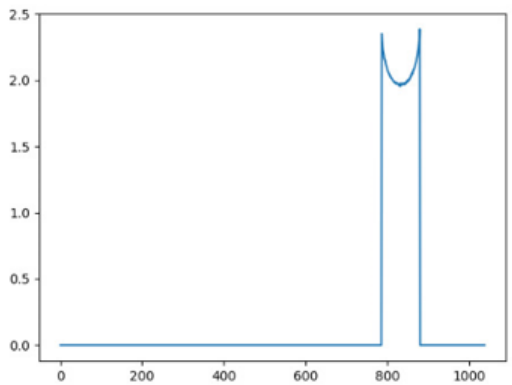

Figura 6.9: Gráfica con las medidas obtenidas en Lidar2d

Esta solución propuesta del cilindro se trata simplemente de una forma de lograr simular la aplicación de seguimiento de una persona, ya que en un entorno real, el láser si que es perfectamente capaz de detectar a las personas, pero al no ser posible en este caso se quiere dejar reflejado como se especula que sería el funcionamiento de haber logrado obtener medidas directamente del modelo del "actor".

Para la obtención del ángulo en el que se encuentra la persona, se ha obtenido la medida más cercana obtenida y se ha analizado en qué posición del vector se encuentra, de tal forma que si de las 1040 medidas obtenidas por este sensor nuestra medida se encuentra en la posición 278 por ejemplo,simplemente se deberá multiplicar el valor de 0.25 grados que es la distancia entre cada medida por 278, obteniendo así con precisión un valor que será el ángulo por el que el objeto se aproxime, empezando siempre a contar por la izquierda como se ha mencionado en el desarrollo.

Para un correcto seguimiento y toma de decisiones a la hora de avanzar u orientarse, el robot divide la totalidad de 260 grados en un total de 4 "sectores", siendo los primeros 115 grados uno, los 30 siguientes otro y los últimos 115 grados el tercer sector tal como se ha visto en la figura 5.4

Con estos sectores, se determina la dirección en la que el robot debe moverse, ya que si la medida obtenida se encuentra en el primer sector, el robot girará hacia la izquierda hasta que dicha medida ya no este dentro de este sector. Lo mismo ocurrirá con el tercer sector, en el cual se emplea el mismo razonamiento que en el primero pero empleando un giro hacia la derecha.

A la hora crear las funciones de movimiento, se observó el problema de que el valor que en el teclado está definido como "k" para detener el robot realmente no presenta ningún valor de velocidades, por lo que se creó un vector para darle a esta variable "k" un valor necesario para detener el robot.

En un principio, se trató de emplear la función de movimiento de avance + giro, pero como ya se ha comentado esta función de movimiento era algo más inestable, provocando constantes caídas.

Finalmente, se empleará el segundo sector para avanzar en línea recta, ya que se supone que el elemento a seguir se encuentra delante del robot.

Una vez aprendido a avanzar y orientarse hacia la persona, se deberán crear ciertas medidas de seguridad para evitar la colisión directa con el actor, puesto que por el momento solo avanzamos sin tener en cuenta ninguna distancia de seguridad.

Por tanto, todos los movimientos anteriores se realizarán única y exclusivamente si la medida más cercana detectada por el robot es superior a las 1.3 unidades mencionadas anteriormente, de lo contrario este no se moverá para evitar cualquier posible colisión, puede resultar una medida algo excesiva, pero debido al pequeño retraso que puede originarse a la hora de enviar ordenes al robot se ha acabado aumentando bastante.

Adicionalmente, se ha creado una distancia de seguridad en la cual el perro se apartará si algo se acerca más de lo que debe, con esto se pretende simular como el robot se apartaría en caso de que el usuario de la vuelta, dicha función funciona de modo que si el usuario se acerca por la derecha o izquierda, este girará y retrocederá rápidamente, mientras que si se acerca de frente simplemente retrocederá evitando cualquier tipo de colisión.

Para facilitar la compresión se dejan a continuación una serie de procedimientos:

En un primer paso, el robot detecta al actor en el que sería el primer sector descrito

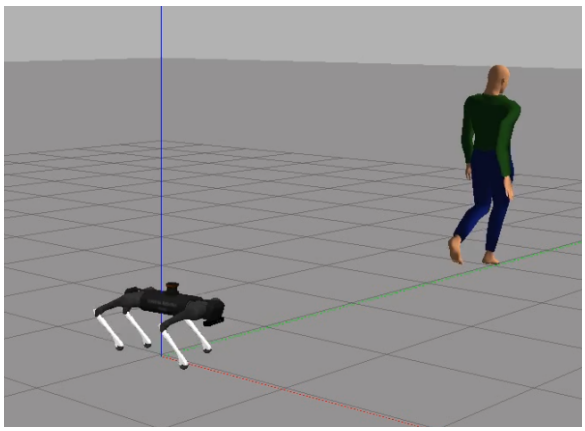

Figura 6.10: Primer paso en simulación Lidar2d

anteriormente y por ello iniciará un giro hacia la izquierda hasta que las lecturas del actor estén en el segundo sector o sector central.

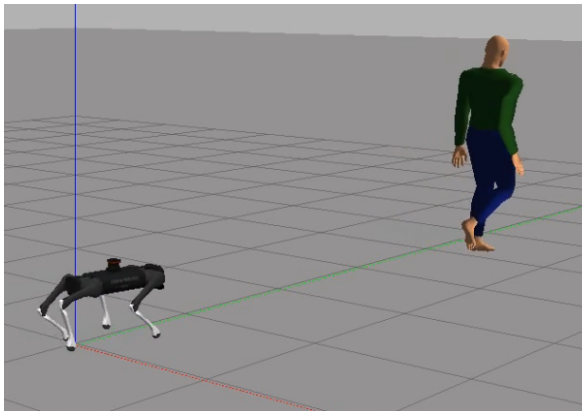

Figura 6.11: Segundo paso en simulación Lidar2d

En un segundo paso, como se puede observar, el robot ya está mirando en la dirección del actor y posteriormente comenzará a caminar hacia el puesto que se respetan las distancias de seguridad y este se encuentra en el segundo sector.

En este tercer paso, el actor cambia su trayectoria, comenzando a moverse hacia la derecha, por lo que el robot deberá detenerse y orientarse de nuevo como en el primer paso, repitiendo el proceso.

Una vez se ha orientado y acercado lo suficiente el actor inicia un cuarto paso, donde se genera un movimiento de nuevo hacia la izquierda, por tanto el robot estará dentro de su trayectoria, por ello primero se detendrá al detectar que el actor está cerca y retrocederá cuando este se acerque tal como se ve en la imagen, donde la pata derecha trasera ya está comenzando el movimiento de retroceso.

Como quinto paso, el robot detecta que el actor se acerca por el que sería su tercer sector (su derecha) sin respetar las distancias de seguridad, por ello retrocederá mientras gira hacia

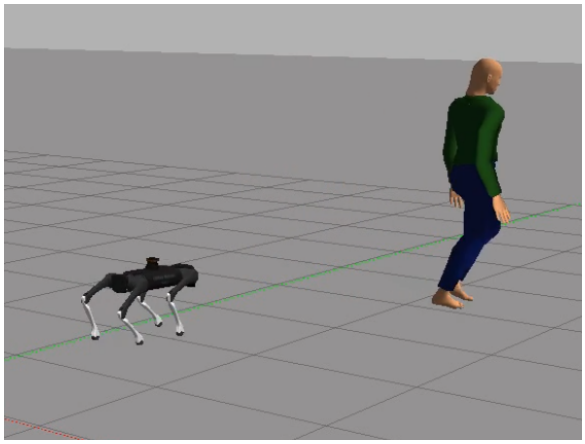

Figura 6.12: Tercer paso en simulación Lidar2d

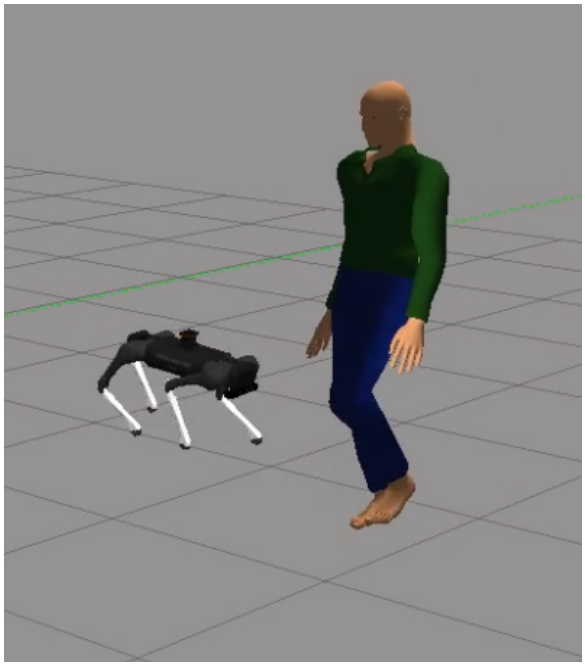

Figura 6.13: Cuarto paso en simulación Lidar2d

la izquierda, logrando así que el actor pueda pasar sin tener que modificar su trayectoria.

Finalmente, el robot volverá a ubicarse en la dirección del actor y continuará siguiéndole sin ningún tipo de problema.

Como a fin de cuentas no se ha llegado a detectar a un humano en un entorno no identificado, se trató de lograr este objetivo empleando el método de detección de DR-SPAAM-Detector, con el cual se deberían de detectar personas cuando estas se muevan en el entorno, para ello se hizo uso del paquete creado por el VisualComputingInstitute sin llegar a obtener unos buenos resultados al tratar de unir ambos proyectos, ya que aparecía un error el cual impedía continuar con este.

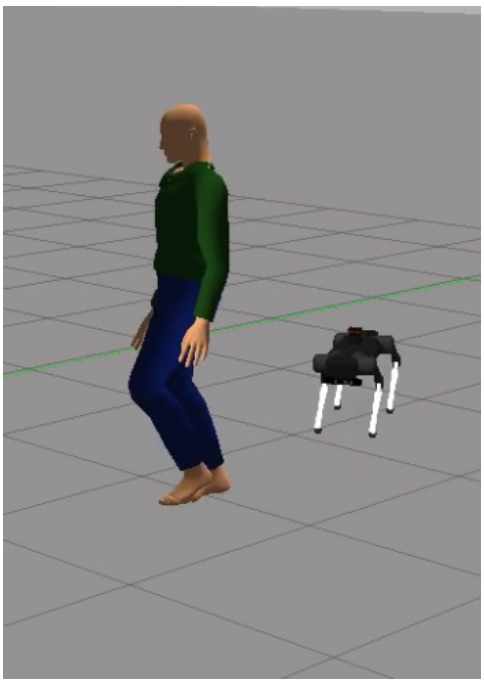

Figura 6.14: Quinto paso en simulación Lidar2d

Aquí se puede observar un pequeño vídeo con la aplicación desarrollada con el fin de facilitar su compresión, en este se puede visualizar el terminal en todo momento para así apreciar que movimiento está decidiendo el robot en cada instante `https://youtu.be/Rveb0AJkMy8`

6.4 Detección de personas con cámara

Una vez se ha controlado el robot mediante el Lidar2d, se tratará de hacer que este funcione de forma autónoma siguiendo a personas empleando la cámara con un algoritmo de reconocimiento en tiempo real.

Con todo esto, se procede a elegir un algoritmo de detección, eligiendo el algoritmo conocido como YOLO (you only look once), el cual como bien indica su nombre es muy rápido haciéndolo ideal para la detección en tiempo real necesaria para controlar el robot, por otra parte, este algoritmo presenta una fiabilidad muy alta además de proporcionar datos de la clase del objeto detectado (que en este caso el objeto es persona), además de proporcionar información sobre la ubicación del objeto mediante las coordenadas de este en la imagen y la probabilidad de que ese objeto se corresponda con el detectado.

Una vez conocido esto, se seleccionará el paquete de darknet ros, con el que aplicar la red YOLO a una imagen publicada en un topic, de este modo, se modificará el topic para que coincida con el de la cámara, cuando la imagen sea detectada con el paquete se mostrarán las imágenes, ya que se trata de un vídeo con la clasificación realizada y mostrando cuadros delimitadores alrededor de las distintas figuras que se encuentren.

Sin embargo, para el control del robot no emplearemos directamente estas imágenes, sino que emplearemos los distintos topics que este paquete crea, principalmente el topic "/darknet

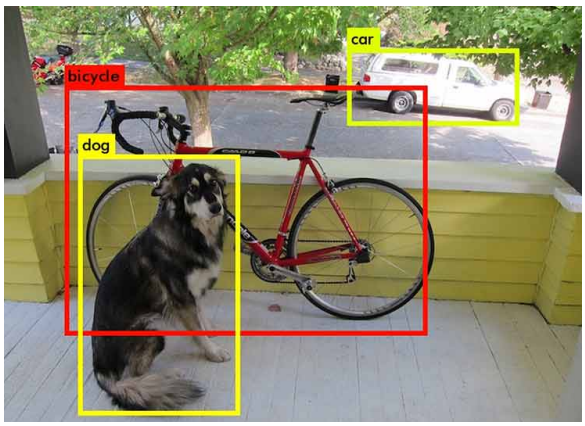

Figura 6.15: Ejemplo de imagen procesada con YOLO

ros/Boundingboxes" del que podremos obtener una lista con todos los elementos deseados, siendo estos los siguientes:

1. Probabilidad
2. Xmin
3. Ymin
4. Xmax
5. Ymax
6. Class

De estos datos, se obtiene la clase además de las coordenadas de dos vértices opuestos del rectángulo que localizan al objeto detectado, empleando una simple media entre las coordenadas X de estos puntos se obtiene aproximadamente la coordenada X dentro de la imagen donde está situado el actor, por tanto con esta información se puede mover el robot teniendo en cuenta que la imagen presenta una resolución de 640x640 píxeles se dividirá al igual que en el Lidar2d la imagen en 3 sectores, un sector de los primeros 200 píxeles, un segundo central de 240 y un tercero de 200 resultando en algo similar a la imagen 6.16

De estas secciones, podemos determinar que si el punto medio descrito anteriormente se sitúa en el primer sector, el robot girará hacia la derecha, si está en el tercero hacia la izquierda y finalmente si se sitúa en la segunda sección (la central) el robot avanzará hacia el objetivo puesto que esto significará que se encuentra frente a el.

En cuanto al resto de datos obtenidos del topic, la probabilidad y las coordenadas Y no se utilizan, ya que no presenta ninguna utilidad la coordenada Y al no poder inclinar el robot ni hacer mucho más que avanzar hacia delante o hacia atrás, con respecto a la probabilidad en un principio nos es indiferente, ya que parece devolver siempre valores superiores al 95

Con estos datos, se modificará el paquete champ teleop con el fin de utilizar las mismas órdenes de movimiento que se usaban durante la teleoperación con teclado, pero dependiendo de los datos obtenidos de los topics de darknet ros Dentro del paquete, se creará un nuevo

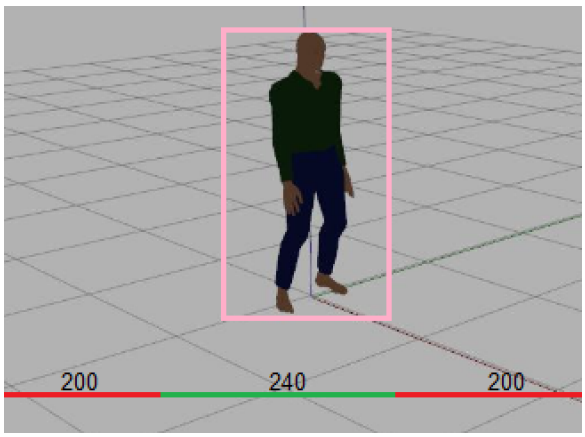

Figura 6.16: Esquema división en secciones de la cámara

script similar al de la teleoperación, sustituyendo la función encargada de leer la tecla pulsada por dos funciones callback que empleando variables globales se encargarán de determinar la dirección en la que el robot debe moverse.

Se encontró otro problema relacionado con la detención del robot cuando este deja de detectar a la persona, ya que en un principio se pretende que el robot se detenga al dejar de detectar a alguien, sin embargo, cuando se deja de detectar a la persona, el topic de darknet deja de publicar mensajes, por lo que no llegamos a obtener un mensaje con 0 personas con el que detenernos en la función callback creada, es por ello que se empleará otra función callback con el fin de modificar la orden de movimiento, de lo contrario si la persona sale de la imagen por la derecha el robot estará permanentemente girando hacia la derecha hasta que encuentre a otra clase persona, en un futuro se estudiará si queremos que la orden cuando deja de ver a la persona es esa pero por el momento no nos es útil, ya que en el caso de dejar de reconocer a una persona que está frente a él porque aparece un obstáculo o cualquier cosa este seguirá en línea recta y acabará colisionando con dicho obstáculo que impedía la visión.

Para solucionar este problema, se creó otro subscriber a un topic de darknet, en este topic se leerá el número de objetos detectados y si este resulta ser 0, se puede asegurar que el topic darknet ros/BoundingBoxes estará inactivo, por tanto se sobre escribirá la variable global encargada del movimiento con el valor ”k”, de este modo se consigue que el robot se detenga al dejar de detectar personas hasta que vuelva a detectarlas, si el objetivo fuese intentar encontrar a una persona a toda costa, sería posible cambiar ese valor ”k” por ”j” o ”l”, logrando así que el robot gire sobre si mismo buscando de nuevo a la persona.

Para garantizar la seguridad del robot, surge otro problema, ya que este se avanza sin parar hacia la persona pero no tenemos forma de detenernos empleando solo la cámara, ya que en un principio la aplicación solo empleará uno de los dos sensores, de tal modo que se pensó en un sistema que hace que el robot se detiene cuando el cuadrado con el que se ha detectado a la persona sea lo suficientemente grande.

Este razonamiento surge del pensamiento de que cuanto más nos acerquemos a la persona más grande será esta en pantalla, pudiendo así crear un área máxima que si la superamos el robot se detendrá. Al igual que en el sensor Lidar2d, se podría hacer que este retroceda pero resulta más complejo, puesto que si el robot se acerca demasiado al actor cuando este avanza

hacia el robot, llega un momento en el que el actor está demasiado cerca del robot y la red neuronal no es capaz de detectar a la persona viendo solamente las piernas de esta.

A continuación, se deja un ejemplo práctico de los resultados obtenidos en la cámara del robot y razonando lo que ocurrirá o está ocurriendo.

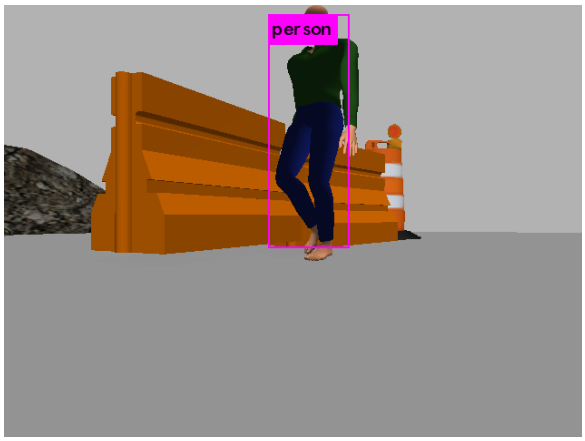

Figura 6.17: Resultado Yolo a larga distancia

En el caso de la figura 6.17, la persona se encuentra lejos, por lo que podremos avanzar con total normalidad hacia ella sin encontrar ningún tipo de error, previamente el robot ha girado en la dirección necesaria para que el actor se sitúe en el centro de la imagen

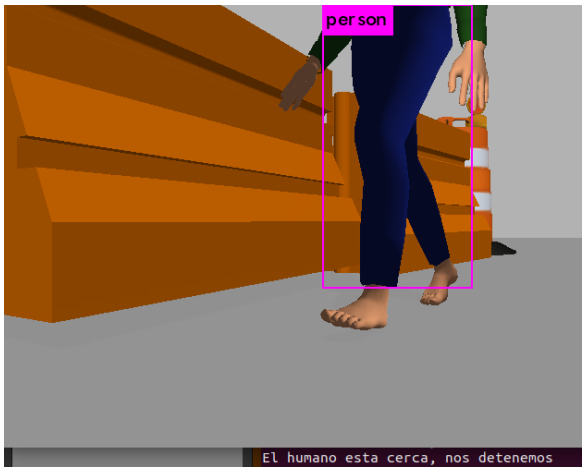

Figura 6.18: Resultado Yolo a corta distancia

Una vez nos hemos acercado, el rectángulo es más grande y con ello su área, por tanto el robot se detiene tal como se puede leer en la terminal situada bajo la imagen 6.18

Utilizando la teleoperación, se ha situado al robot aún más cerca del actor, hasta tal punto que no reconoce la clase persona en él empleando la red neuronal de yolo, por tanto no es posible saber si estamos demasiado cerca al no tener información alguna, este resultado se puede apreciar en la imagen 6.19

Figura 6.19: Problema al usar Yolo a una distancia demasiado corta

Por ello, se probó a emplear otra versión de yolo, llamada YOLOv3 sin la versión tiny, la cual acelera la red,obteniendo unos mejores resultados, sin embargo su coste computacional es demasiado elevado para el hardware con el que estoy trabajando (8Gb de RAM y una 1060 de 6Gb) por lo que no se terminó de indagar en este método y se dejó simplemente la medida que detiene al robot cuando el área supera dicho umbral.

Para finalizar con esta sección, se han reflejado los resultados obtenidos en vídeos al igual que con Lidar, en estos se pueden apreciar los distintos errores mencionados además de otros como puede ser la confusión de algún elemento con una persona, los vídeos están en el siguiente enlace: `https://youtube.com/playlist?list=PLgvdh4WIJFUtNZlbcdUU_PXofh4vgtqLS`. La baja tasa de fotogramas en Gazebo es debida al elevado coste que presenta mover el robot mientras que empleamos el paquete de darknet ros.

6.5 Mejoras empleando Lidar2D

Debido a los resultados obtenidos con el sensor Lidar2D, se ha decidido retomar este, tratando de mejorar la aplicación obtenida en la sección 6.3. Para ello, se ha utilizado el algoritmo de detección de personas conocido como DR-SPAAM-Detector con el cual se diferencian los obstáculos de las personas en función de si estos se mueven o no, para ello se ha empleado el directorio del VisualComputingInstitute , en el cual se facilita un nodo en ROS con el cual al introducir mensajes con la estructura ”sensor msgs/LaserScan” este obtiene una interfaz donde se visualizan los puntos detectados por el sensor destacando aquellos que puedan ser personas con una pequeña señal roja.

Sin embargo, tras instalar reiteradas veces el repositorio y algunas librerías que son requeridas en python, no se ha logrado que este funcione, por tanto queda pendiente para su uso en un futuro, puesto que no se ha llegado a obtener resultado alguno con él.

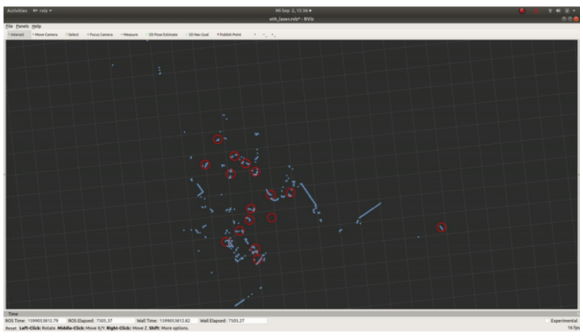

Figura 6.20: Representación ejemplo obtenida en Rviz con el detector SPAAM

7 Conclusiones

Tras haber terminado el proyecto, podemos llegar a la conclusión de que los objetivos planteados en un principio han sido superados con éxito, sin embargo me gustaría destacar el descontento con la detección empleando el Lidar2d, ya que desde un principio se pensó en emplear dicho sistema de detección directamente con el actor, aun así el resultado obtenido a pesar de los problemas encontrados ha sido satisfactorio, ya que durante el desarrollo del proyecto se han ido afrontando distintos problemas y estos se han resuelto en su totalidad con éxito, aumentando los conocimientos y la compresión del entorno de Gazebo y las distintas comunicaciones que podemos emplear para obtener de nuestra simulación en este caso los datos que deseemos así como modificarlos según deseemos, ya que también se han desarrollado gracias a esto los conocimientos sobre Python obtenidos a la hora de jugar con los distintos topics, además de algunos conocimientos obtenidos en C++, ya que aunque no se llegó a obtener un resultado óptimo con él desde un principio se realizaron algunas aplicaciones que se descartaron.

Aun con esto, la principal satisfacción a nivel personal es la capacidad de facilitar a cualquier persona que quiera simular su propio robot cuadrúpedo y así poder desarrollar su aplicación del mismo modo que he hecho yo a lo largo del proyecto, ya que la obtención de un paquete con el cual poder controlar correctamente cualquier robot cuadrúpedo con tan solo disponer de su Urdf ha sido una de las tareas más complicadas debido a la ausencia de recursos y a los errores que suelen aparecer al emplear cualquiera de estos.

8 Trabajos Futuros

Para terminar, me gustaría reflejar que este proyecto no está ni mucho menos acabado, ya que a partir de su modificación se pueden llegar a obtener infinidad de aplicaciones, de entre las cuales se pretende profundizar en el empleo de este sistema para la creación de perros lazarillo, ya que empleando un láser del mismo tipo pero apuntando en dirección contraria podríamos saber si la persona a la que pretendemos guiar sigue detrás nuestra; mientras que, con la cámara y un GPS o algún sistema que se planteé en el futuro se podría crear un sistema capaz de llevar al robot a la ubicación deseada mientras este emplea todos los sensores para llegar a ese lugar garantizando que su acompañante le sigue y evitando cualquier tipo de peligro que afecte tanto a él mismo como a su acompañante.

En cuanto a lo realizado, también se podría indagar en la forma de dotar a los modelos de los actores de físicas u obtener algún sensor láser el cual si que sea capaz de detectarlos con el fin de terminar la aplicación correctamente, puesto que los datos obtenidos de un cilindro nunca serán los mismos que los obtenidos por dos piernas humanas. Destacar además la posibilidad de profundizar más en la red neuronal de Yolo, ya que esta ha sido empleada en su versión tiny debido a que de lo contrario el hardware empleado no era el mejor para la tarea, aun así los resultado obtenidos con la red cabe destacar que son más que satisfactorios.

En cuanto a formas de mejorar los métodos descritos en este proyecto, en un futuro se mejorará la detección con el Lidar2D, ya que considero que este tipo de sensor puede tener un gran potencial empleando algoritmos de detección como es el Dr Spaam, con el cual no se ha llegado a indagar del todo y creo que puede dar mucho juego en el futuro, logrando así la implementación que no se ha llegado a desarrollar en este proyecto.

Bibliografía

Anjali Mukherjee, K. I. R., S. Adarsh. (2020). *Ros-based pedestrian detection and distance estimation algorithm using stereo vision, leddar and cnn.*

Bellegarda, G. (2021). *Robust high-speed running for quadruped robots via deep reinforcement learning.*

Elena Garcia, J. E., Pablo Gonzalez de Santos. (2006). *Quadrupedal locomotion. an introduction to the control of four-legged robots.*

Hamed Bozorgi, H. M. L., Xuan Tung Truong, y Ngo, T. D. (2021). *2d laser and 3d camera data integration and filtering for human trajectory tracking.*

Hiroshi Kimura, A. H. C., Yasuhiro Fukuoka. (2007). Adaptive dynamic walking of a quadruped robot on natural ground based on biological concepts. *The International Journal of Robotics*(1), 16.

Jia, D., Hermans, A., y Leibe, B. (2020). DR-SPAAM: A Spatial-Attention and Autoregressive Model for Person Detection in 2D Range Data. En *International conference on intelligent robots and systems (iros).*

Jiannan Chen, Z. S., Ping Ye. (2019). *Pedestrian detection and tracking based on 2d lidar.*

Petrus Sutyasadi, M. P. (2015). *Trotting control of a quadruped robot using pid-ilc.*

Prajesh Raveendran, A. T. J. R., Siva Ariram. (2020). *Development of task-oriented ros-based autonomous ugv with 3d object detection.* IEEE.

PriyaranjanBiswa, P. K. (2021). Development of quadruped walking robots: A review. *Ain Shams Engineering Journal*, *12*(1), 2017-2031.

T.T.K Dat, P. T. T. (2014). *A study on locomotions of quadruped robot.*

Yuto Nagai, Y. T., Yuya Sawano. (2022). *A method for detecting human by 2d-lidar.*

Lista de Acrónimos y Abreviaturas

2D	2 Dimensiones.
CPU	Central processing unit.
FPS	Frames per second.
GB	Gigabyte.
GPS	Global Positioning System.
GPU	Graphics processing unit.
HOG	Histograma de Gradientes Orientados.
IEEE	Institute of Electrical and Electronics Engineers.
LIDAR	Light Detection and Ranging.
TFG	Trabajo Final de Grado.
YOLO	You Only Look Once.