Escuela
Politécnica
Superior

Planificación y control de la marcha de un exoesqueleto de miembro inferior mediante visión

Grado en Ingeniería Robótica

Trabajo Fin de Grado

Autor:
Carlos Eduardo Fernández García
Tutor/es:
Jorge Pomares Baeza

Universitat d'Alacant
Universidad de Alicante

Enero 2023

Planificación y control de la marcha de un exoesqueleto de miembro inferior mediante visión

Autor
Carlos Eduardo Fernández García

Tutor/es
Jorge Pomares Baeza
Departamento de Física, Ingeniería de Sistemas y Teoría de la Señal

Grado en Ingeniería Robótica

Escuela
Politécnica
Superior

Universitat d'Alacant
Universidad de Alicante

ALICANTE, Enero 2023

Preámbulo

En este trabajo se pretende desarrollar un sistema de detección de escaleras mediante visión y la generación de trayectorias para que un exoesqueleto de miembro inferior pueda subirlas. La decisión de realizar este proyecto reside la cantidad de personas, no solo de la tercera edad, que hoy en día y en el futuro tendrán problemas de movilidad debido a diferentes causas y que requerirán de medios para poder desplazarse de manera autónoma. Con este trabajo se pretende facilitar a estas personas la tarea de subir escaleras empleando un exoesqueleto de miembro inferior y así poder acceder a lugares en los que no hay alternativa. De este modo se pretende dotar de mayor libertad a personas con problemas de movilidad.

Agradecimientos

Este trabajo no hubiera sido posible sin el apoyo y ayuda de muchas personas, por lo que quisiera dedicarles este trabajo y compartir con ellos el mérito.

Quisiera dedicarle el trabajo a mi familia, ya que han creído en mi incluso más que yo mismo, me han convertido en la persona que soy, me han dado la libertad de decidir mi futuro y con su esfuerzo y dedicación me han dado todo lo que he necesitado y más para hacerme fácil estos años.

Quisiera dedicarle el trabajo a mi pareja, la persona que más me ha ayudado a levantarme cada vez que me he caído y la persona que más me ha apoyado y aconsejado en cada paso que he dado.

Quisiera dedicarle el trabajo a mis compañeros del grado, debido a que me han hecho reír cada día de clase, me han acompañado en mis logros y sobretodo en mis fracasos, ayudándome ante múltiples dificultades.

Por último, quisiera dedicarle también el trabajo a mi tutor Jorge Pomares, que me dio la oportunidad de formar parte de un proyecto que me ilusionó y con el que disfruté cada día, que me ha dado la libertad de realizar el trabajo que he deseado y que me ha aconsejado en cada duda que me ha surgido.

A todas estas personas y a las que se me habrá olvidado mencionar, gracias.

Da tu primer paso con fe.
No tienes por qué ver toda la escalera.
Basta con que subas el primer peldaño.

Martin Luther King.

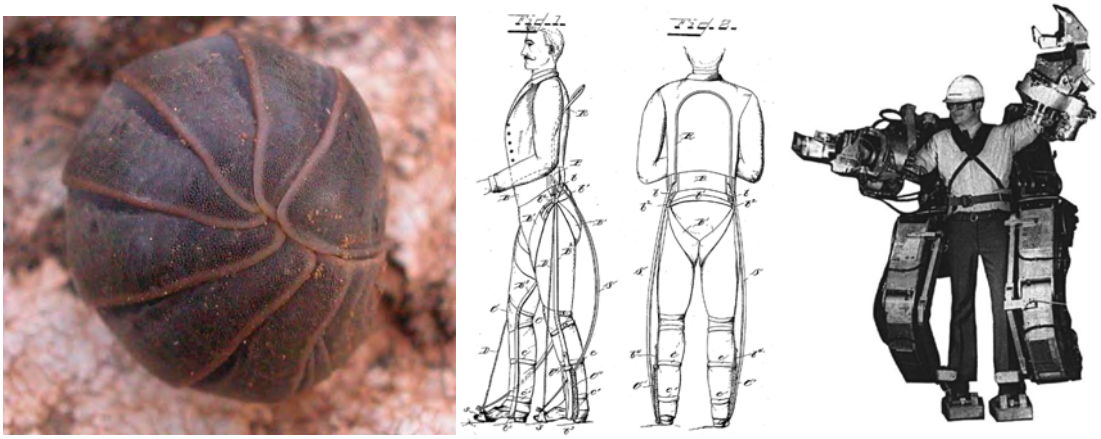
Índice general

Índice de figuras

Índice de cuadros

1 Introducción

El término “exoesqueleto” fue inicialmente empleado para describir la estructura biológica que aporta protección y resistencia a órganos blandos en el interior del cuerpo de algunos animales (Figura 1.1 A). En 1890 se patentó el primer exoesqueleto para humanos (Figura 1.1 B) que consistía en un dispositivo pasivo que facilitaba la realización de diferentes actividades como caminar, correr y saltar, Yang (1890). La concepción que se tiene en la actualidad del exoesqueleto proviene del proyecto que se desarrolló en 1965 entre el Departamento de Defensa de los Estados Unidos y la empresa General Electric. El proyecto consistió en la creación del exoesqueleto Hardiman, un prototipo concebido para levantar pesos de hasta 682 kg (Figura 1.1 C), aunque para el 1970 únicamente se desarrolló un brazo que podría levantar 341 kg Bogue (2009). En la actualidad, los exoesqueletos se relacionan con dispositivos robóticos portátiles que realizan o ayudan a realizar actividades a sus usuarios, Bao y cols. (2019).



(a) Exoesqueleto biológico Garcia (2008)

(b) Exoesqueleto pasivo Yang (1890)

(c) Hardiman Bogue (2009)

Figura 1.1: Origen y evolución de los exoesqueletos

Los exoesqueletos pueden emplearse para tareas desde mejorar la fuerza hasta usarse como prótesis. Desde su creación, han evolucionado en diferentes sectores como el militar, el industrial o el médico entre otros. Un ejemplo en el sector militar es el desarrollado por la Agencia de Proyectos de Investigación Avanzados de Defensa (DARPA), en colaboración con empresas privadas, las cuales están desarrollando un exoesqueleto desde el año 2000 denominado “El soldado del mañana” (Figura 1.2 A). Este exoesqueleto proporciona la capacidad de levantar repetidamente 90 kg aproximadamente sin agotar al usuario.

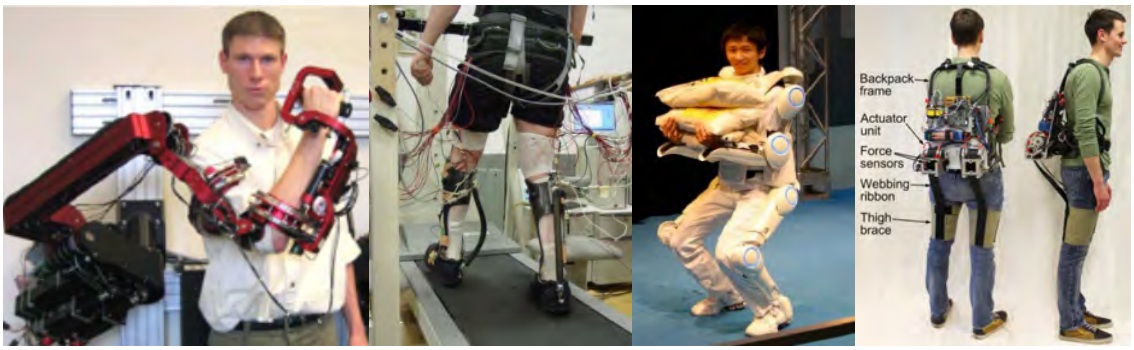
En el sector médico se puede encontrar un proyecto desarrollado por la Universidad de Agricultura y Tecnología de Tokyo que consiste en un exoesqueleto destinado a ayudar a los

agricultores con tareas físicamente exigentes (Figura 1.2 B) como arrancar cultivos, labrar la tierra o podar árboles. Este proyecto tiene una gran importancia debido a que en Japón casi la mitad de agricultores tienen 65 años o más, por lo que la realización de determinados trabajos pueden producir dolencias y enfermedades.

(a) "El soldado del mañana" **(b)** Exoesqueleto para agricultores

Figura 1.2: Comparación de exoesqueletos diseñados para diferentes sectores Bogue (2009)

Dependiendo de las partes del ser humano en las que participan, los exoesqueletos se pueden clasificar en 4 tipos:



(a) Miembro superior Lo y Xie (2012)

(b) Miembro inferior Huo y cols. (2016)

(c) Cuerpo completo Huo y cols. (2016)

(d) Articulación específica Asbeck y cols. (2015)

Figura 1.3: Clasificación de los exoesqueletos según la zona del cuerpo humano en la que actúan

Los exoesqueletos serán cada vez más necesarios ya que el envejecimiento de la población es un problema global debido a que los humanos cada vez viven más años. El aumento de la esperanza de vida es debido a múltiples factores como los avances de la medicina. Al alcanzar edades avanzadas, las personas pueden padecer deterioros físicos, por lo que esta situación se ha convertido en un problema socio-económico para multitud de países. Una encuesta de Naciones Unidas reveló que el 11,5% de la población en 2012 tenía más de 60 años y que para el 2050 sería más del doble .

La fragilidad en las personas de avanzada edad es causada por la disminución de masa muscular, lo que conlleva una reducción de actividades que pueden realizar como caminar y a su vez produce una aceleración en el deterioro del sistema neuro-muscular. También hay enfermedades comunes entre las personas de avanzada edad que pueden propiciar la pérdida de movilidad como los accidentes cardiovasculares: trombosis, embolias, etc.

Además de las enfermedades comunes en personas de avanzada edad, también hay enfermedades que se suelen manifestar en pacientes menores de 30 años que causan problemas de movilidad como las lesiones medulares. Los pacientes que las padecen tienen un riesgo elevado de sufrir enfermedades secundarias derivadas como osteoporosis, obesidad, problemas coronarios, diabetes, etc. Por tanto es evidente que la falta de movilidad en pacientes conlleva a una disminución de las expectativas de vida, por lo que sería beneficioso que recuperasen la movilidad de nuevo.

Dependiendo de la función del exoesqueleto de miembro inferior, se pueden clasificar en los siguientes grupos: rehabilitación, locomoción asistencial y potenciación de habilidades físicas (Figura 1.4).

Rehabilitación: Los pacientes que han perdido musculatura pueden no ser capaces de caminar de la misma manera que cuando estaban sanos o haber perdido estabilidad. En la rehabilitación tradicional, el terapeuta proporciona a los pacientes un entrenamiento en el que el especialista ejecuta junto al convaleciente unos ejercicios de manera repetitiva para que, tras un período de tiempo, se recuperen o reduzcan sus dolencias. Estos ejercicios suelen producir un desgaste físico en el terapeuta y suelen ser ineficientes. Los exoesqueletos diseñados para la rehabilitación realizan el trabajo físico, ejecutando los ejercicios repetidamente, por lo que liberan al terapeuta del esfuerzo físico y les facilita centrarse en el rendimiento del paciente. Además los exoesqueletos pueden realizar y registrar movimientos con fuerzas y velocidades concretas gracias a los sensores que tienen incorporados.

Locomoción asistencial: Los exoesqueletos para la locomoción asistencial son principalmente empleados para ayudar a pacientes con parálisis que han perdido completamente la movilidad en las extremidades inferiores. Proporcionan el torque externo necesario en las articulaciones para reemplazar al del usuario. Estos exoesqueletos proporcionan la habilidad de realizar actividades diarias como levantarse, sentarse y caminar , Chen y cols. (2016).

Este trabajo se va a centrar en los exoesqueletos de miembro inferior y en las funciones de rehabilitación y locomoción asistencial.

(a) Rehabilitación **(b)** Locomoción asistencial **(c)** Potenciación de habilidades físicas

Figura 1.4: Clasificación exoesqueletos de miembros inferiores Chen y cols. (2016)

2 Motivación y Objetivos

El objetivo principal de este trabajo es que un exoesqueleto de miembro inferior sea capaz de subir escaleras. Para ello, en este trabajo se pretende desarrollar un sistema para la extracción de las dimensiones de una escalera mediante una cámara de profundidad y cargar dichas dimensiones en el simulador TOWR, junto con el modelo del exoesqueleto, para generar trayectorias correspondientes. Para lograr el objetivo principal será necesario completar una serie de subobjetivos:

- Manejo de la cámara de profundidad Intel RealSense L515 LiDAR desde ROS para realizar capturas del entorno, controlando el número de *frames* que se desean capturar de cada escena y publicar la información capturada en *topics*.
- Capturar nubes de puntos de barreras arquitectónicas como escaleras reales, tanto de interior como de exterior, variando parámetros como la altura de la cámara para determinar los valores óptimos.
- Procesar las nubes de puntos capturadas para extraer las dimensiones de los escalones y compararlos con las medidas reales de la escalera para determinar la precisión de la extracción.
- Diseñar el modelo del exoesqueleto en Xpp, a partir un modelo 3D, para que se puedan visualizar los movimientos mientras se desplaza por el terreno con una posición y orientación determinadas por la trayectoria generada por TOWR.
- Incorporar el modelo 3D del exoesqueleto a TOWR para que el simulador planifique las trayectorias ajustándose a la capacidad de movimiento.
- Replicar las escaleras en el simulador TOWR con las dimensiones extraídas de las nubes de puntos para proporcionarle al simulador un entorno sobre el que obtener las trayectorias superando obstáculos.
- Configurar los parámetros del simulador para que genere y ejecute correctamente las trayectorias necesarias para alcanzar una posición objetivo superando los obstáculos replicados en el terreno.

3 Marco Teórico

En este apartado se va a realizar un análisis de diferentes trabajos relacionados con este proyecto, de los cuales se comentarán los apartados que tengan que ver con el mismo o se consideren relevantes. De este modo se obtendrá un base de información que servirá para determinar, justificar y evaluar el desarrollo del trabajo.

3.1 Detección de escaleras

Para el problema de que un robot suba unas escaleras, se pueden plantear soluciones desde diferentes enfoques. Mientras que algunos trabajos se centran en el hardware como método para poder subir escaleras Tantichattanont y cols. (2007), otros trabajos se centran en la detección y medición de escaleras como tarea clave para subirlas, Luo y cols. (2013). Este último trabajo se centra más concretamente, en el desarrollo de un método de reconocimiento y medición de escaleras en tiempo real para identificarlas en entornos de interior. A continuación se entrará más en detalle para explicar el método desarrollado por Luo y cols. (2013).

Los robots diseñados para subir escaleras peldaño a peldaño requieren un alto nivel de percepción de las escaleras, sin embargo dicha percepción normalmente se ve afectada por el movimiento de los sensores al desplazarse, por lo que no se puede garantizar resultados muy precisos mientras el robot las está subiendo. Existen diferentes métodos que se pueden combinar para detectar escaleras. El principal problema de estos métodos es que requieren de un alto coste temporal para obtener resultados, lo que puede imposibilitar el correcto funcionamiento del sistema en tiempo real. En el trabajo de Luo y cols. (2013) realizan el reconocimiento de escaleras haciendo uso de métodos estadísticos de datos 3D. Para el sistema de percepción emplearon una cámara Kinect y una IMU. La cámara Kinect se empleó para capturar la escena y la IMU para predecir el ángulo de visión de la cámara. El sistema se dividió en 5 apartados: procesado de la nube de puntos (Kinect), determinación del ángulo de visión (IMU), detección de los planos de la escalera, ajuste del modelo de escalera y emparejamiento del modelo. En la siguiente figura se puede ver un esquema de los 5 apartados.

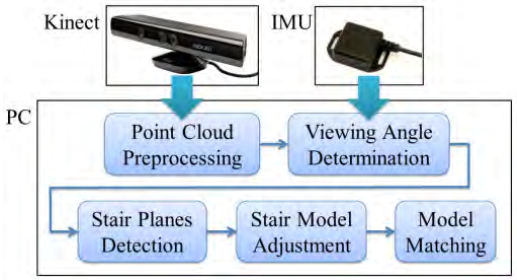


Figura 3.1: Esquema representativo de los 5 apartados del sistema implementado en Luo y cols. (2013).

El trabajo solo tiene en cuenta las situaciones en las que la cámara está situada a una cierta altura y apuntando a la escalera con un cierto ángulo hacía abajo, de modo que en la escena siempre habrá una escalera. Para detectar los planos que conforman una escalera se realiza una búsqueda de las direcciones verticales observando las distribución de los vectores normales. Se definen dos tipos de vectores normales $\mathbf{n}_{i,j}$ y $\mathbf{n'}_{i,j}$ por cada punto $\mathbf{p}$ en las coordenadas de píxeles (i,j) perteneciente a la nube de puntos $\mathbf{P}$. Los vectores normales se calculan mediante las siguientes expresiones:

$$\mathbf{v}_i = p(i,j) - \mathbf{p}(i + \Delta i, j), v_j = \mathbf{p}(i,j) - \mathbf{p}(i, j + \Delta j) \tag{3.1}$$

$$\mathbf{v}'_i = \mathbf{p}(i,j) - \mathbf{p}(i + k\Delta i, j), v'_j = \mathbf{p}(i,j) - \mathbf{p}(i, j + k\Delta j) \tag{3.2}$$

$$\mathbf{n}_{i,j} = \mathbf{v}_i x \mathbf{v}_j, \mathbf{n}'_{i,j} = \mathbf{v}'_i x \mathbf{v}'_j \tag{3.3}$$

Donde Δi y Δj se corresponde al desplazamiento de los píxeles de las filas y columnas, cuyos valores han sido determinados como 1, y k es un pequeño integrador con un valor mayor de 1. Además, $\mathbf{n}_{i,j}$ se corresponde con un vector normal de un área pequeño, mientras que $\mathbf{n}'_{i,j}$ se corresponde con un vector normal de un área mayor para un mismo punto. De este modo, para encontrar los vectores normales que se encuentran en planos, se comparan los valores obtenidos de cada vector normal y si son similares pertenecerán a un plano. La expresión para comprar los vectores normales es la siguiente:

$$\frac{(n_{i,j})}{|(n_{i,j})|} \cdot \frac{(n'_{i,j})}{\left|(n'_{i,j})\right|} \geq n_{th} \tag{3.4}$$

Donde el umbral n_{th} tiene un valor comprendido entre 0.9 y 1. Una vez determinados los planos, se agrupan según la altura o nivel en el que se encuentren. Para ello se asume que la dirección vertical de la nube de puntos se corresponde con el eje z, el cual apunta hacia arriba. En el histograma de la distribución de vértices de los puntos 3D (Figura 3.2) se pueden encontrar algunos picos definidos haciendo uso de la siguiente expresión:

$$max\{N_{i-2}, N_{i-1}, N_i, N_{i+1}, N_{i+2}, N_{th}\} = N_i \Leftrightarrow i \in H \tag{3.5}$$

Donde N_i se corresponde con la acumulación del número de puntos 3D con su valor "z" en el rango del índice "i". N_{th} es un umbralizado para filtrar pequeños picos. H es el conjunto de índices de picos que representan los posibles planos horizontales.

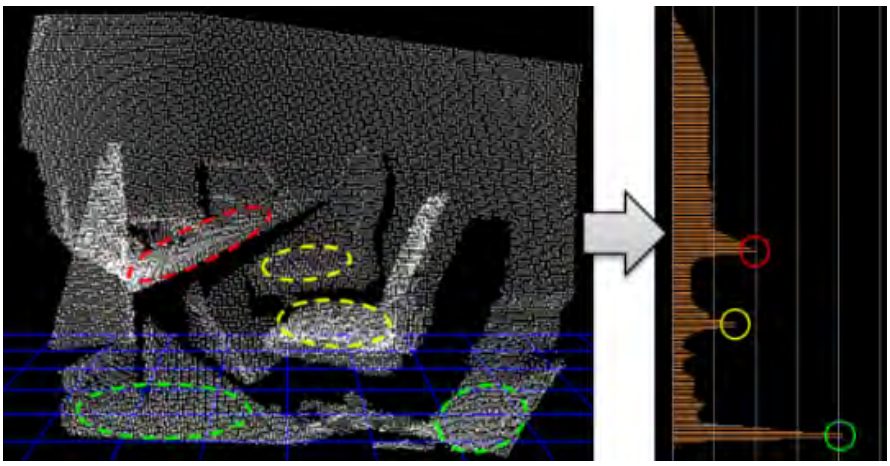


Figura 3.2: Detección de diferentes planos en una escena mediante un histograma. Cada color representa una altura distinta Luo y cols. (2013).

En el trabajo de Luo y cols. (2013), los planos horizontales son la clave para determinar donde hay escaleras y donde no. Las escaleras están generalmente compuestas por una serie de planos horizontales equidistantes con la profundidad y el ancho adecuados. El inicio de los planos horizontales se corresponde con los bordes, los cuales son la característica más importante para que un robot pueda subir una escalera. En la siguiente imagen (Figura 3.3) se pueden ver los parámetros que definen una escalera.

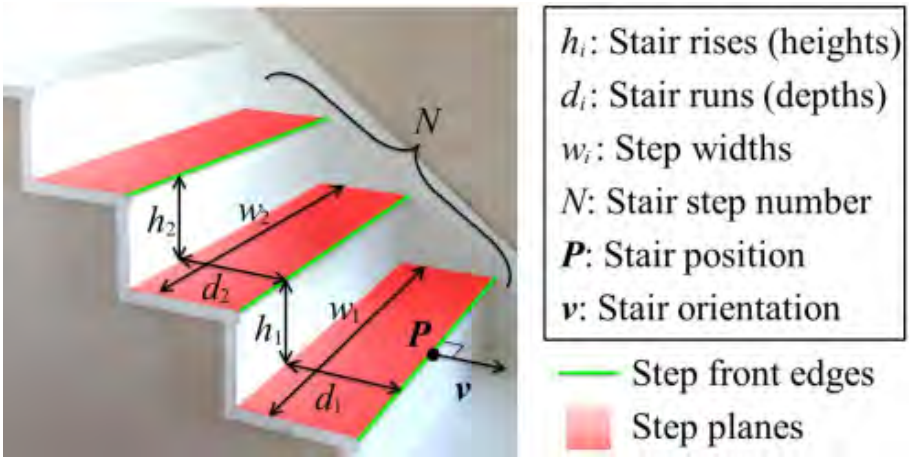


Figura 3.3: Parámetros que definen una escalera Luo y cols. (2013).

El trabajo de Luo y cols. (2013) también trata el tema de la rectificación del ángulo de visión, ya que la cámara está a cierta altura apuntando hacia abajo con un cierto ángulo. El ángulo se calcula mediante la siguiente expresión:

$$\mathbf{g} + \mathbf{a}_k = a_x \mathbf{e}_x + a_y \mathbf{e}_y + a_z \mathbf{e}_z \tag{3.6}$$

Donde $\mathbf{a}_k$ se corresponde con la aceleración de la cámara Kinect, a_x, a_y, a_z son los valores de la aceleración en los 3 ejes principales ($\mathbf{e}_x$,$\mathbf{e}_y$ y $\mathbf{e}_z$). Generalmente, la dirección de la gravedad es difícil encontrarla sin la información de $\mathbf{a}_k$, sin embargo en entornos de interior es poco probable que los robots realicen movimientos con aceleraciones muy altas $|\mathbf{a}_k|$, por lo que se puede asumir que el valor de $|\mathbf{a}_k|$ es bastante menor que el de $|\mathbf{g}|$ y que la dirección de $\mathbf{g'} = \mathbf{g} + \mathbf{a}_k$ es similar a la dirección de $\mathbf{g}$. Por tanto, ya que solo se requiere una aproximación de la dirección vertical, se puede tomar la dirección de $\mathbf{g'}$ como $\mathbf{g}$.

Tras esta aclaración, se rotan todos los puntos de la nube para rectificar el ángulo de la cámara. Se pueden encontrar todos los posibles planos horizontales a partir del histograma de posición vertical de los puntos 3D. De todos los posibles planos, se escogen los que tengan propiedades semejantes de espaciado como probables superficies de los escalones. De este modo y realizando una serie de ajustes, se obtienen las superficies de los escalones donde el robot se apoyará para subir la escalera.

El trabajo de Luo y cols. (2013) también comenta los resultados de las 5 pruebas experimentales que se realizaron. A continuación se muestra una tabla con los errores de la altura de los escalones ($\mathbf{h}$) de cada una de las pruebas:

Escalera	A	B	C	D
Error de h (cm)	0.3± 0.4	0.3±0.3	-0.2±0.3	0.4±0.5

Cuadro 3.1: Errores en la determinación de la altura de los escalones ($\mathbf{h}$)

Por último el trabajo de Luo y cols. (2013) concluye que aunque los errores de las observaciones solo están en un rango de ±2.0 cm, la media de los 5 experimentos se puede reducir a un rango de ±0.4 cm. Debido a que cada experimento tiene una duración de 0.1 segundos, los 5 seguidos tendrían una duración de 0.5 segundos, lo que sería suficientemente rápido y preciso para tareas de subir escaleras.

Otra forma de detectar escaleras para que que un robot las pueda subir es el que se propone en el trabajo Oßwald y cols. (2011), en el que abordan el problema de construir un modelo 3D con escaleras complejas, como por ejemplo escaleras con giros, basándose en datos adquiridos mediante un láser situado en la cabeza del robot humanoide Nao. Estos modelos de extracción de planos requieren de la suficiente precisión para lograr que el robot suba escaleras. Oßwald y cols. (2011) evalúan dos enfoques del estado del arte. El primero agrupa líneas de exploración de la escena, mientras que el segundo estima las direcciones principales de la escena mediante puntos de muestreo al azar y los agrupa para encontrar planos ortogonales a las direcciones principales. Estos métodos han funcionado correctamente en la navegación sobre superficies planas como el suelo.

Para que los robots humanoides puedan funcionar de manera autónoma y desplazarse libremente por entornos diseñados para personas, deben ser capaces de subir escaleras. Para ello requieren de modelos de escaleras precisos.

El primer método fue presentado en el artículo *3D Perception and Environment Map Generation for Humanoid Robot Navigation* por Gutmann, Gutmann y cols. (2008). Consiste

en la agrupación y extracción de líneas y segmentos de líneas vecinas en una imagen de rango. Los segmentos de líneas son combinados formando segmentos de planos si pertenecen a un mismo plano.

El segundo método fue desarrollado por Kida en el artículo *Human finding and body property estimation by using floor segmentation and 3D labelling*, Kida y cols. (2004). Se basa se basa en el muestreo aleatorio de dos puntos. La idea principal consiste en estimar las direcciones principales del entorno mediante puntos aleatorios y determinar las diferencias entre sus vectores. Posteriormente se agrupan para encontrar segmentos de planos que son ortogonales con las direcciones principales.

En el trabajo se expone que, según la experiencia de los autores, métodos como RANSAC tienden a simplificar demasiado las estructuras planas convirtiendo pequeños escalones en rampas planas.

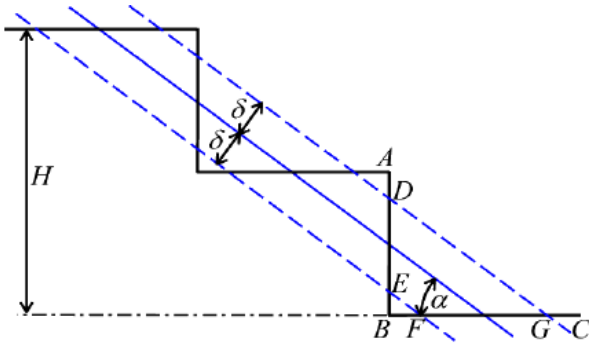


Figura 3.4: Ejemplo del posible error al realizar una simplificación del perfil de una escalera al de una rampa, Qian y Ye (2014).

En el proceso de adquisición de la nube de puntos hay varios aspectos y limitaciones que se han de tener en cuenta. El primero se trata de que, estando el láser sobre la cabeza del robot a unos 64 cm de altura, hay una distancia horizontal mínima de 50 cm en el que el sistema no es capaz de percibir lo que hay (zona ciega), esto se debe al ángulo máximo que se puede mover el láser. Esta limitación se puede ver representada en la siguiente figura:

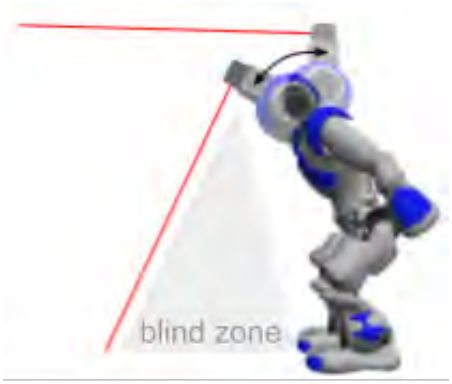


Figura 3.5: Representación el rango de visión del sensor láser sobre el robot Nao y de la zona ciega, Oßwald y cols. (2011).

El segundo aspecto que se debe tener en cuenta es el ruido causado por la imprecisión del propio láser, así como la estimación de la pose del sensor a partir de la cinemática directa en función del ángulo con el que se inclina el láser. El efecto del ruido se puede ver en la siguiente figura:

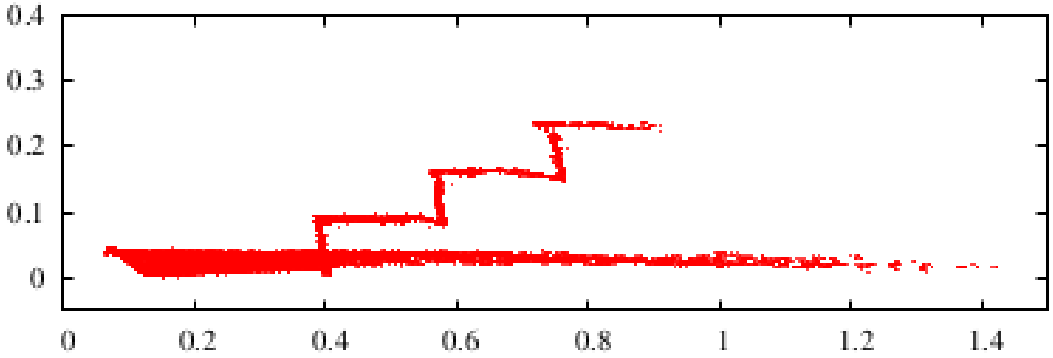


Figura 3.6: Perfil de la escalera en el que se puede apreciar el ruido en la adquisición de la nube de puntos, Oßwald y cols. (2011).

A continuación se van a comentar los dos métodos de extracción de planos que se proponen para la detección de una escalera.

El primer método inicialmente se realiza un escaneado del entorno. Seguidamente se ajustan las líneas de segmentos para cada fila de la imagen, para ello se dividen los puntos en grupos, de modo que un grupo nuevo empieza cuando se excede una distancia entre vecinos de 5 cm. Para cada grupo se calcula una línea recta mediante mínimos cuadrados. A continuación se evalúa la posición de los puntos respecto de la línea. Si hay más de un número determinado de puntos consecutivos (15) a un lado de la línea, el grupo se reduce al primer y último punto. Las líneas con menos puntos que un valor determinado (5) o menor a un umbral (5 cm) son descartadas. La extracción de los planos se realiza encontrando 3 segmentos de líneas en las filas vecinas de la imagen, las cuales se superponen en el inicio y final de sus columnas. Además, las 3 líneas deben pasar un test estadístico para ser consideradas como posibles planos. El algoritmo finaliza cuando no se pueden encontrar más posibles planos.

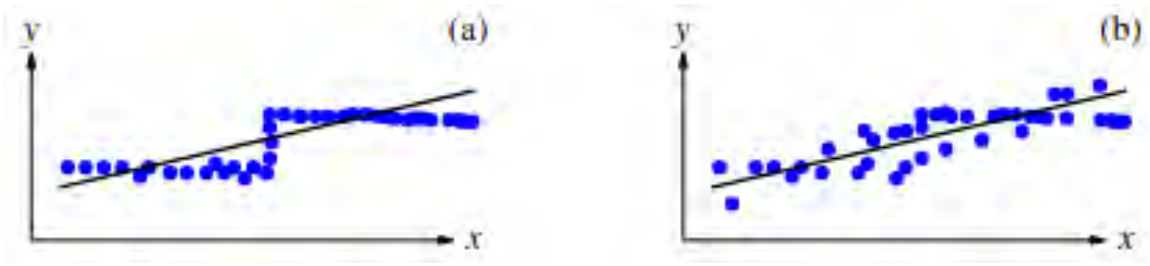


Figura 3.7: Evaluación de los puntos respecto a una línea. (A) datos precisos, (B) datos distorsionados, Gutmann y cols. (2008).

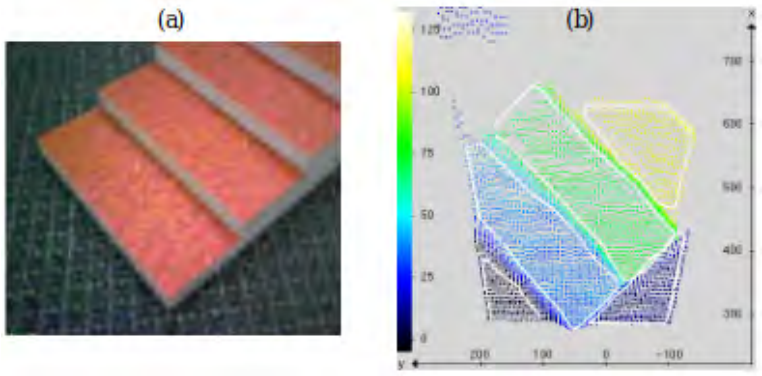


Figura 3.8: Detección de planos aplicando el primer método, Gutmann y cols. (2008). (A) imagen original, (B) datos de rango segmentados

El segundo método se basa en que los entornos creados por las personas, muchas de las superficies están alienadas con un conjuntos de direcciones principales. El algoritmo inicialmente determina las direcciones principales de la nube de puntos mediante un muestreo. A continuación, corta los datos en rebanadas perpendiculares a cada dirección principal y busca segmentos del plano dentro de cada rebanada. Dado un conjunto de puntos $\mathbf{p}_1$, ... , $\mathbf{p}_N$ el algoritmo extrae los segmentos del plano siguiendo los siguientes pasos:

1. Muestreo de un conjunto V de diferencia de vectores normalizados. Los elementos de V pueden ser interpretados como puntos en una esfera unitaria.

$$V \leftarrow \left\{ \frac{\mathbf{p}_i - \mathbf{p}_j}{|\mathbf{p}_i - \mathbf{p}_j|} para\,(1 \leq i, j \leq N, i \neq j) \right\} \tag{3.7}$$

   Los puntos $\mathbf{p}_i$ y $\mathbf{p}_j$ proceden de diferentes superficies del entorno generados aleatoriamente distribuidos en la esfera unitaria. Si $\mathbf{p}_i$ y $\mathbf{p}_j$ se originan desde misma superficie plana, indicará que el punto correspondiente en V se encuentra en el círculo unitario con el mismo vector normal que la superficie. Este es el motivo por el que los puntos dibujados desde superficies similares producen anillos con una alta densidad de puntos en una esfera unitaria. Considerando la densidad de puntos se puede distinguir las superficies planas del ruido.

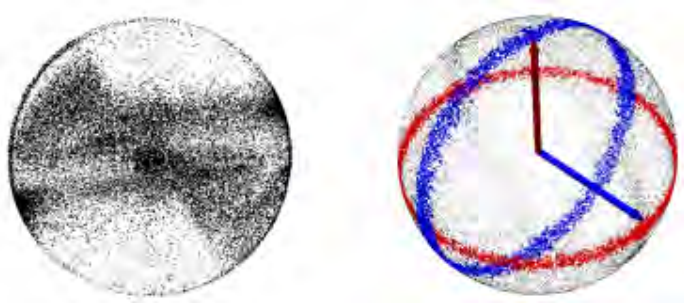


Figura 3.9: Círculo unitario con la extracción de las direcciones principales, Kida y cols. (2004)

2. Búsqueda del grupo en forma de anillo R = $\{v_1, ..., v_M\} \subseteq$ V

3. Determinación del vector normal de R:

$$\mathbf{n} \leftarrow \arg\max_{\mathbf{n}} \sum_{k=1}^{M} \begin{cases} 1 & \text{if } \mathbf{n}^T \mathbf{v}_k \approx 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$

4. Se corta la nube de puntos en finas rebanadas perpendiculares a **n**

5. Para cada rebanada que contenga un número suficiente de puntos (20) se agrupan dichos puntos dentro de los segmentos de corte de cada plano. Para encontrar más planos se han de repetir los pasos 2 y 5.

Para construir las escaleras con la información obtenida y procesada, se intersectan los planos horizontales y verticales para obtener las líneas horizontales que contienen los bordes de los escalones. A partir de los bordes y con los datos de las superficies verticales y horizontales se puede reconstruir la escalera.

Ambos métodos son suficientemente precisos para subir una escalera de manera autónoma. La complejidad computacional del primer método (agrupación de líneas de exploración) es significativamente más eficiente debido a que trabaja sobre segmentos de líneas extraídas de la imagen de rango, mientras que el segundo método (basado en muestreo) trabaja con toda la nube y requiere un gran número de iteraciones para encontrar todos los segmentos de planos. En la siguiente tabla se pueden comparar los resultados que se obtuvieron.

	Método 1	Método 2
Valores Escalón	Agrupación de líneas de exploración	Muestreo de dos puntos aleatorios
Altura (7 cm)	0.42±0.31 cm	0.68±0.54 cm
Anchura (60 cm)	3.40±1.95 cm	2.25±1.97 cm
Profundidad (18 cm)	1.17±0.67 cm	0.90±0.61 cm
Planos paralelos	2.22±2.17 º	1.14±1.13 º
Ángulos 90 º	4.97±2.13 º	3.12±1.47 º
Tiempo de ejecución	**0.025±0.001 s**	**3.102±1.043 s**

Cuadro 3.2: Comparación del error y tiempo de ejecución entre los dos métodos propuestos en el trabajo *From 3D Point Clouds to Climbing Stairs: A Comparison of Plane Segmentation Approaches for Humanoids*, Oßwald y cols. (2011).

Por último, se va a comentar el trabajo Woo y cols. (2019), debido a que, además de realizar la detección de unas escaleras, se trata el tema de transformaciones geométricas del entorno capturado para el procesado de la escena.

Desde hace años, muchos trabajos de investigación han tratado el tema de la detección de escaleras. La información que se obtiene del mapeado del entorno es utilizada para colocar correctamente los pies del robot. A diferencia de los robots humanoides, los robots cuadrúpedos tienden a llevar el sensor a una altura baja, por lo que no pueden ver varios escalones en una misma escena.

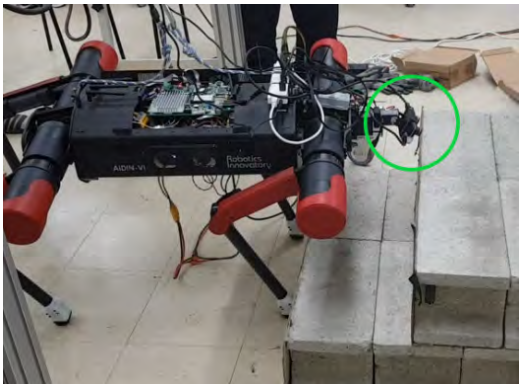
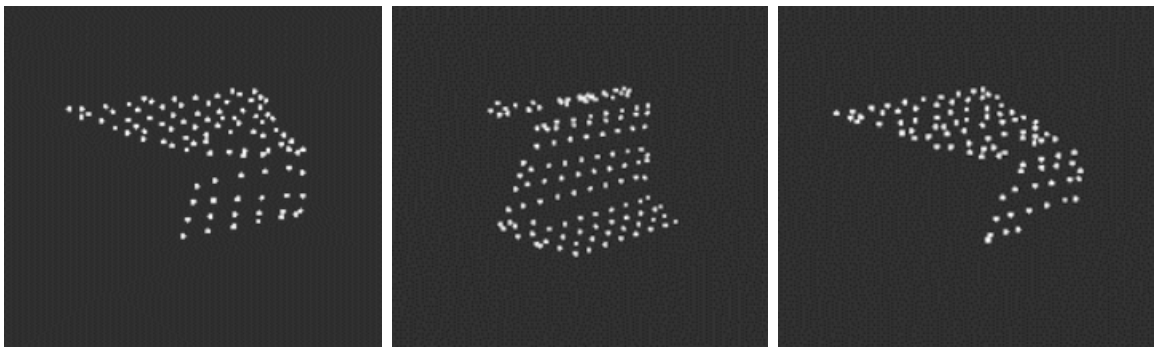


Figura 3.10: Posición de la cámara de profundidad en un robot cuadrúpedo. En la figura se puede apreciar que en la posición en la que se encuentra no puede detectar el primer escalón, Woo y cols. (2019)

Debido al movimiento del robot, las escenas se mueven rápidamente aún cuando el robot se mueve lentamente, lo que conlleva a una distorsión de la escena, por lo que emplear únicamente la odometría por visión es inexacto. Por ello en este trabajo se centran en el desarrollo de un algoritmo para robots cuadrúpedos con el objetivo de navegar por entornos con escaleras haciendo uso de una cámara de profundidad y de un sensor IMU. La característica fundamental del algoritmo se basa en que se seleccionará el mejor plano de entre los posibles planos que forman un escalón.

Inicialmente, se emplea el algoritmo RANSAC para reducir el número de puntos. Seguidamente se seleccionan conjuntos aleatorios de 3 puntos, conformando un plano, y se cuenta el número de puntos que contiene (*inlier points*). Si dicho número supera un umbral predefinido (número de puntos que debe contener un plano), el plano será seleccionado como un plano real. Una vez que se selecciona como plano real se eliminan los puntos que se encuentran contenidos y se repite el procedimiento sin dichos puntos. El algoritmo se repite hasta que el número de puntos restantes en la nube sea menor al umbral predefinido de número de puntos que ha de tener un plano.

Para conseguir una mayor precisión, de cada escena se pueden tomar varias capturas y comparar los resultados. A cada captura de una misma escena se le denomina *frame*. Para una misma escena, aunque la posición de la cámara no se modifique, la disposición de los puntos en cada frame es diferente. Este hecho se vuelve aún más complicado cuando la cámara se mueve, ya que conforme se modifica la posición de la cámara, la escena va cambiando.



(a) Captura 1 **(b)** Captura 2 **(c)** Captura 3

Figura 3.11: Captura de una misma escena desde posiciones diferentes,Woo y cols. (2019)

Para solucionar este problema se realizan unas trasformaciones de las coordenadas de la cámara al sistema de coordenadas global. Para realizar las transformaciones se emplean las siguientes expresiones:

$$X = (x, y, z, 1)^t \tag{3.9}$$

$$X' = TX \tag{3.10}$$

Donde X representa el punto en el sistema de coordenadas de la cámara y X' en el sistema de coordenadas global. T representa la matriz de transformación 4x4 del sistema de la cámara al sistema global.

$$P = (a, b, c, d)^t \tag{3.11}$$

$$P' = (T^{-1})^t P \tag{3.12}$$

P representa los coeficientes del plano en el sistema de la cámara y P' del sistema global. Cabe destacar que únicamente se transforman los puntos contenidos en el plano (inlier points). De este modo, en el sistema de coordenadas global, los mismos planos provenientes de diferentes frames se colocan en casi las mismas posiciones, haciendo posible su comparación.

Figura 3.12: Visualización de los diferentes planos, provenientes de varios frames, una vez transformados al sistema de coordenadas global, Woo y cols. (2019)

La distancia entre dos planos horizontales paralelos se corresponderá con la altura del escalón (d) y de modo similar, la distancia entre dos planos verticales paralelos se corresponderá con la profundidad del escalón (p). Para determinar si dos planos son paralelos se pueden comparar sus ángulos a partir del cálculo del ángulo del vector normal.

Posteriormente se realiza una clasificación de los planos extraídos de cada frame en los diferentes escalones de la escena. De este modo, para cada escalón se agrupan varios planos provenientes de cada frame.

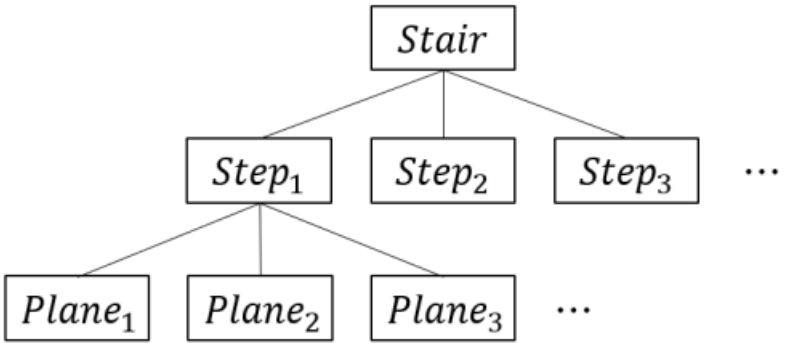


Figura 3.13: Clasificación de los planos en los diferentes escalones que componen la escalera, Woo y cols. (2019)

Tras la clasificación, se obtiene para cada escalón un plano resultante a partir de la media de todos los planos agrupados para dicho escalón.

$$M = \{avg(Step_1), avg(Step_2), avg(Step_3), ...\} = \{M_1, M_2, M_3, ...\} \tag{3.13}$$

M_k representa el el valor de la posición de cada escalón.

Una ventaja de las escaleras es que tienen una estructura muy regular, es por eso que se puede estimar la posición del siguiente escalón a partir de los previos. Para realizar esta estimación se requieren como mínimo dos planos horizontales paralelos para extraer la altura

y dos planos verticales paralelos para extraer la profundidad del escalón. Para realizar dicha estimación se emplea la siguiente expresión:

$$v_k = \frac{\sum_{n=0}^{k} v_n}{k} \tag{3.14}$$

Donde v_n representa el vector normal del plano y v_k el vector normal del plano correspondiente al escalón késimo. Para estimar la altura de los planos se emplea la siguiente expresión:

$$d_k = \frac{k}{k-1} d_{k-1} - \frac{1}{k-1} d_0 \tag{3.15}$$

Donde d_k representa el valor del escalón késimo. La altura se define como la suma de la altura anterior y la media de las alturas de los escalones anteriores.

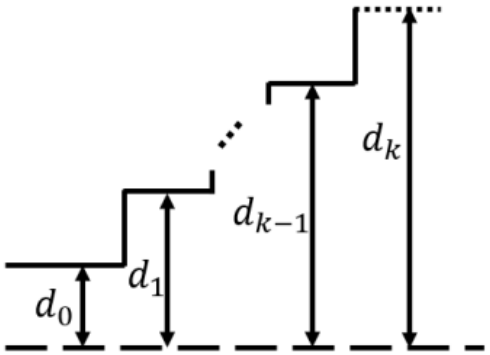


Figura 3.14: Representación del coeficiente d el cual representa la altura de cada escalón, Woo y cols. (2019)

En el trabajo también se comenta que el hardware empleado fue una cámara Intel Realsense d-435 y para el software se empleó ROS

3.2 Generación de trayectorias

La generación de trayectorias para sistemas con patas es difícil debido a que el movimiento se genera a partir de los puntos de contacto de los pies con el entorno, por lo que las fuerzas generadas en dichos puntos deben planificarse cuidadosamente para lograr el comportamiento deseado. En el trabajo de A. W. Winkler y cols. (2018), del que el simulador TOWR basa su funcionamiento, se desarrolla una formulación de optimización de la trayectoria para la locomoción de robots con patas, que determina automáticamente la secuencia de marcha, los tiempos de paso, los puntos de apoyo, los movimientos de balanceo de las piernas y el movimiento corporal 6D sobre un terreno. El sistema se representa utilizando un modelo de dinámica centroidal simplificado que está influenciado por la ubicación y la fuerza de los pies del robot. En función de las características del terreno, se hace cumplir las restricciones del cono de fricción para garantizar que los pies no resbalen. El solucionador de problemas no lineales (NLP) implementado genera planes de movimiento altamente dinámicos con fases de vuelo completas para una variedad de robots con patas con diversas morfologías de manera eficiente. A continuación se va a profundizar en los aspectos relevantes para la comprensión de la formulación descrita en el artículo mencionado.

3.2.1 Problemática

El optimizador de trayectorias (TO), Betts (1998) se puede emplear para generar movimientos de manera general y automatizada, dejando que el usuario únicamente especifique las tareas de alto nivel como la posición inicial y final, dejando que el optimizador determine los movimientos y las fuerzas requeridas para la locomoción cumpliendo una serie de restricciones. El principal problema reside en como trasladar este problema de optimización de tiempo continuo en uno con un número finito de variables de decisión y restricciones para ser resulto por un problema de programación no lineal (NLP).

3.2.2 Modelos Cinemático y Dinámico

El modelo cinemático se determina a partir de la posición de la base y de los efectores finales del robot (pies). Para restringir las posiciones articulares no se emplean directamente unos límites articulares, en su lugar se limitan en función de la posición del pie en el espacio cartesiano. Para ello se define un espacio en el que mientras el pie se encuentre dentro, no se superan los límites articulares.

Por otro lado, en el modelo dinámico, se emplea un modelo de cuerpo rígido simple del que únicamente se requiere la masa e inercia del centro de masas (CoM). La inercia rotacional empleada se calculada a partir de la posición inicial de robot en reposo, lo que denota que las masas de las extremidades son despreciables comparadas con la masa del centro del robot o que las extremidades no se desvían significativamente de su posición por defecto Bernat Iborra (2022).

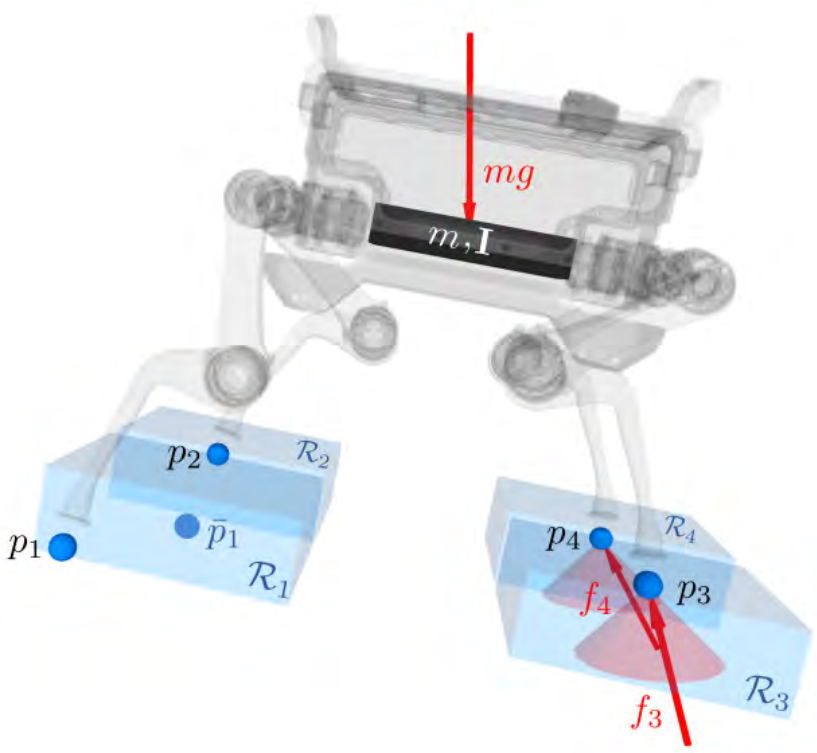


Figura 3.15: Ejemplo del modelo de un robot en TOWR, A. W. Winkler y cols. (2018). Los cubos azules representan los límites articulares del modelo cinemático, mientras que el cubo negro representa el centro de masas del modelo dinámico.

3.2.3 Modelo movimiento

El modelo de movimiento que se genera se basa en patrones de marcha arbitrarios, de los que se modifica las duraciones de las distintas fases que componen el movimiento. Para un robot bípedo, las fases del movimiento serían las siguientes:

- Fase **L**: pie izquierdo en contacto con una superficie.
- Fase **R**: pie derecho en contacto con una superficie.
- Fase **D**: ambos pies en contacto con una superficie.
- Fase **F**: ningún pie en contacto con una superficie, el robot se encuentra en el aire.

Para generar el movimiento deseado, se modifica la duración de las fases de cada pata del robot y se va alternando cada pierna entre la fase contacto con el suelo y la de vuelo. En la Figura 3.16 se pueden apreciar dos patrones de marcha distintos para un robot bípedo. En la primera el robot inicialmente da un paso con el pie izquierdo y tras estar nuevamente en contacto con el suelo, realiza un paso con el derecho, de modo que en todo momento hay un pie en contacto con el suelo. Por el contrario en el segundo patrón de marcha, al modificar la duración de los períodos de las fases, se produce una fase (F) en la que el robot no tiene ningún pie en contacto con el suelo y por tanto está en el aire.

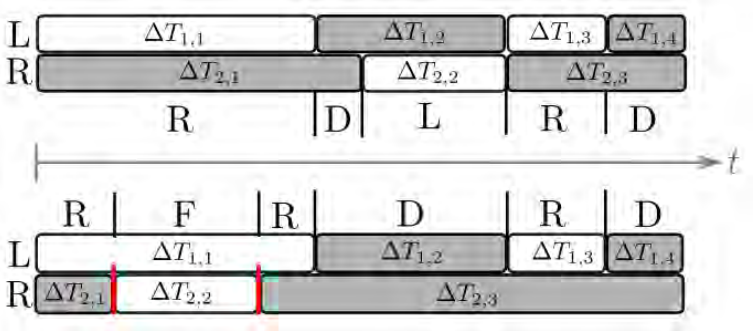


Figura 3.16: Representación de dos patrones de marcha con diferente duración de las fases para un robot bípedo, A. W. Winkler y cols. (2018). El color blanco simboliza el pie en el aire y el color gris en contacto con una superficie.

3.2.4 Algoritmo

El algoritmo recibe como entrada la posición inicial y final deseada del robot, la duración total del movimiento T y el número de pasos $n_{s,i}$ de cada pie i. A partir de estos datos, el algoritmo calcula diferentes trayectorias para la posición lineal del centro de masas $r(t)$, su orientación $\theta(t)$, el movimiento de los pies $p_i(t)$ y la fuerza de contacto $f_i(t)$ para cada pie y determina el un patrón de marcha apropiado definido por $\Delta T_{i,j}$, Bernat Iborra (2022).

find	$\mathbf{r}(t) \in \mathbb{R}^3$	(CoM linear position)
	$\boldsymbol{\theta}(t) \in \mathbb{R}^3$	(base euler angles)
	for every foot i :	
	$\Delta T_{i,1} \ldots, \Delta T_{i,2n_{s,i}} \in \mathbb{R}$	(phase durations)
	$\mathbf{p}_i(t, \Delta T_{i,1}, \ldots) \in \mathbb{R}^3$	(foot position)
	$\mathbf{f}_i(t, \Delta T_{i,1}, \ldots) \in \mathbb{R}^3$	(force at foot)
s.t.	$[\mathbf{r}, \boldsymbol{\theta}](t{=}0) = [\mathbf{r}_0, \boldsymbol{\theta}_0]$	(initial state)
	$\mathbf{r}(t{=}T) = \mathbf{r}_g$	(desired goal)
	$[\ddot{\mathbf{r}}, \dot{\boldsymbol{\omega}}]^T = \mathbf{f}_d(\mathbf{r}, \mathbf{p}_1, \ldots, \mathbf{f}_1, \ldots)$	(dynamic model)
	for every foot i :	
	$\mathbf{p}_i(t) \in \mathcal{R}_i(\mathbf{r}, \boldsymbol{\theta})$,	(kinematic model)
	if foot i in contact :	
	$\dot{\mathbf{p}}_i(t \in \mathcal{C}_i) = \mathbf{0}$	(no slip)
	$p_i^z(t \in \mathcal{C}_i) = h_{terrain}(\mathbf{p}_i^{xy})$	(terrain height)
	$\mathbf{f}_i(t \in \mathcal{C}_i) \cdot \mathbf{n}(\mathbf{p}_i^{xy}) \geq 0$	(pushing force)
	$\mathbf{f}_i(t \in \mathcal{C}_i) \in \mathcal{F}(\mu, \mathbf{n}, \mathbf{p}_i^{xy})$	(friction cone)
	if foot i in air :	
	$\mathbf{f}_i(t \notin \mathcal{C}_i) = \mathbf{0}$	(no force in air)
	$\sum_{j=1}^{2n_{s,i}} \Delta T_{i,j} = T$	(total duration)

Figura 3.17: Algoritmo de generación de trayectorias que emplea TOWR. A. W. Winkler y cols. (2018)

3.2.5 Parametrización de la fases del movimiento

- **Movimiento de la base 6D**: para las variables de posición y orientación ($r(t)$ y $\theta(t)$) se emplean polinomios de cuarto orden y de duración fija encadenados para crear un *spline* continuo y realizar la optimización de sus coeficientes.
- **Movimiento del pie**: ($p_i(t)$) se emplean varios polinomios de tercer orden por cada fase del balanceo y un valor constante para la fase en la que se apoya.
- **Perfil de fuerza del pie**: ($f_i(t)$) se representa cada fase del pie en contacto con el suelo empleando multitud de polinomios, mientras que la fuerza cero se determina durante la fase de balanceo. En la Figura 3.18 se puede ver representado lo anteriormente comentado.

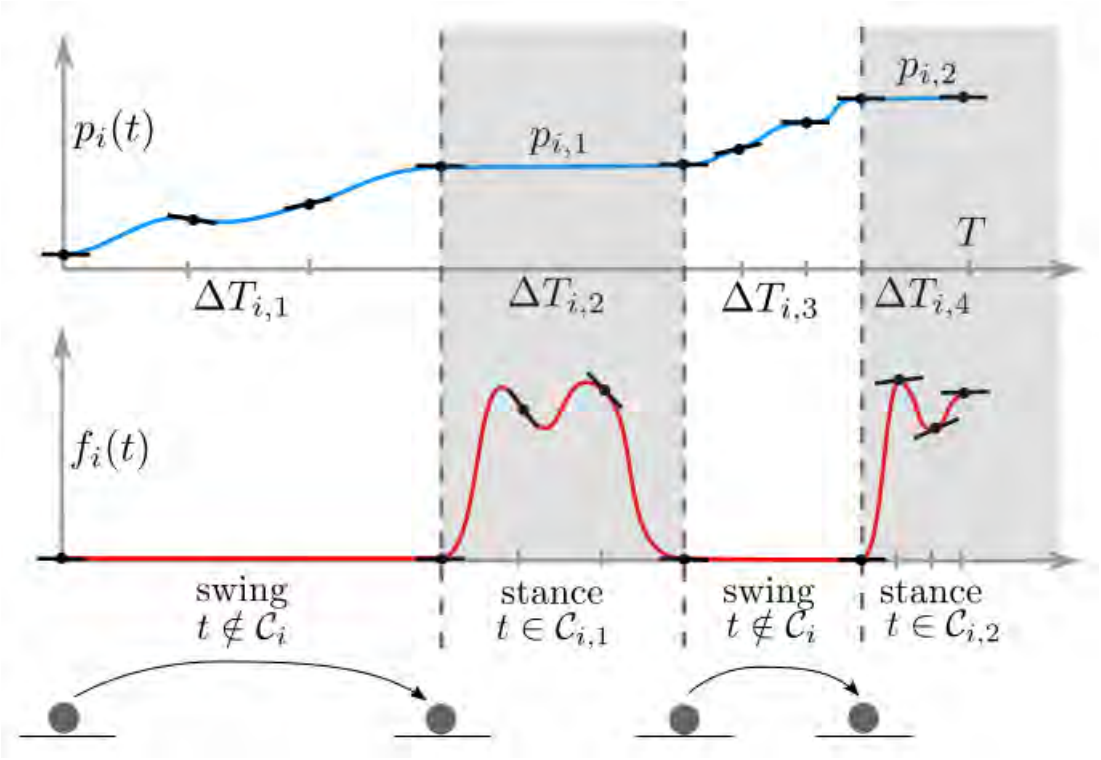


Figura 3.18: Representación de la parametrización de las fases del movimiento, A. W. Winkler y cols. (2018)

4 Metodología

En este apartado se van a enumerar y explicar brevemente todas las herramientas empleadas para la realización de este trabajo. Para justificar el motivo de la utilización de cada herramienta, se ha realizado un esquema visible en la figura 4.1, en el que se detallan las diferentes fases y tareas que se realizan.

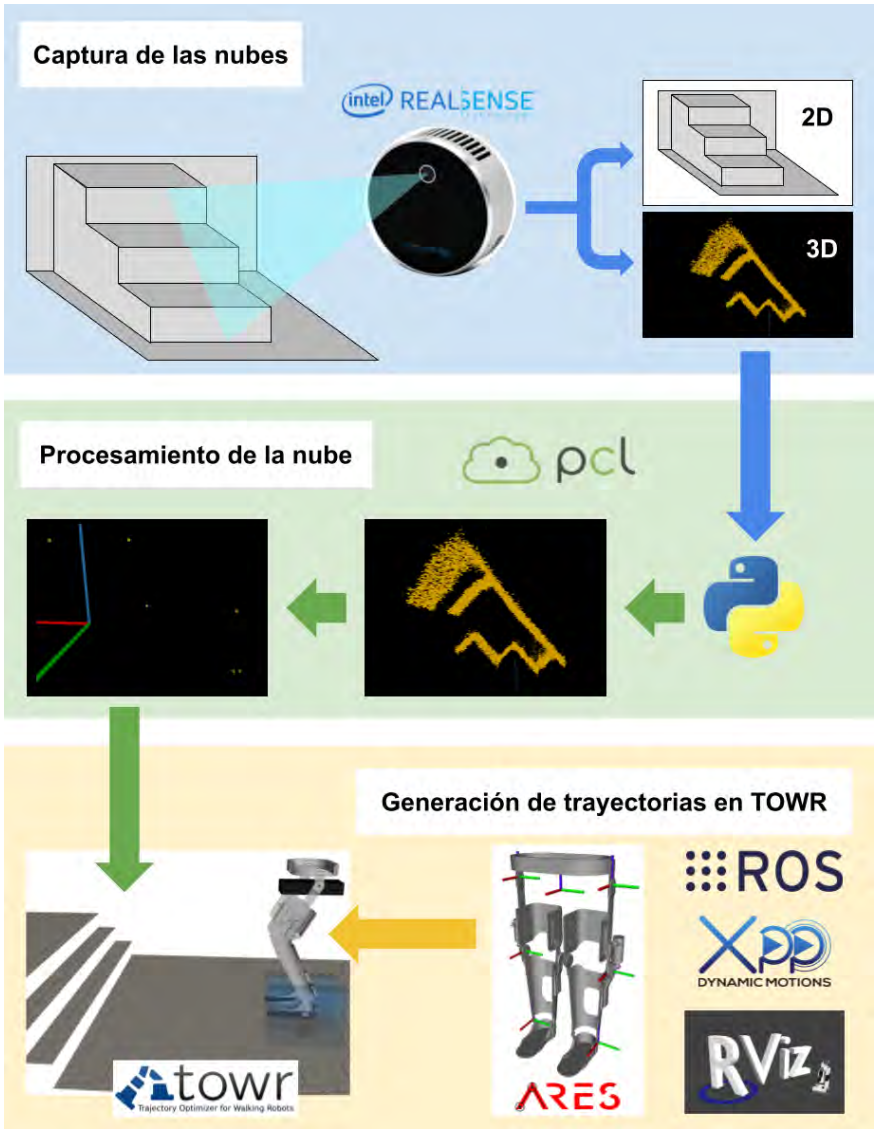


Figura 4.1: Esquema del desarrollo del proyecto.

4.1 Python

Python es un lenguaje de programación de código abierto, alto nivel e interpretado, compatible con todos los sistemas operativos y posee una sintaxis clara y fácilmente legible. Se trata de un lenguaje multiparadigma dado que permite la programación estructurada, funcional y orientada a objetos. Dispone de una biblioteca estándar y multitud de bibliotecas con gran cantidad de funciones implementadas que ayudan y facilitan la implementación de códigos, *Python* (2022).

4.2 Numpy

Numpy es un paquete de Python que se emplea para la computación científica ya que proporciona herramientas como un objeto de matriz multidimensional, objetos derivados como matrices y matrices enmascaradas. También permite realizar operaciones rápidas con matrices, álgebra lineal básica, etc, Harris y cols. (2020).

4.3 PCL

PCL se trata de la librería de código abierto para el procesamiento de nubes de puntos 2D y 3D. Contiene múltiples algoritmos de última generación con los que se pueden realizar tareas de filtrado, estimación de características, reconstrucción de superficies, registro, ajuste de modelos y segmentación. Su uso es gratuito tanto para investigación como para uso comercial, Rusu y Cousins (2011).

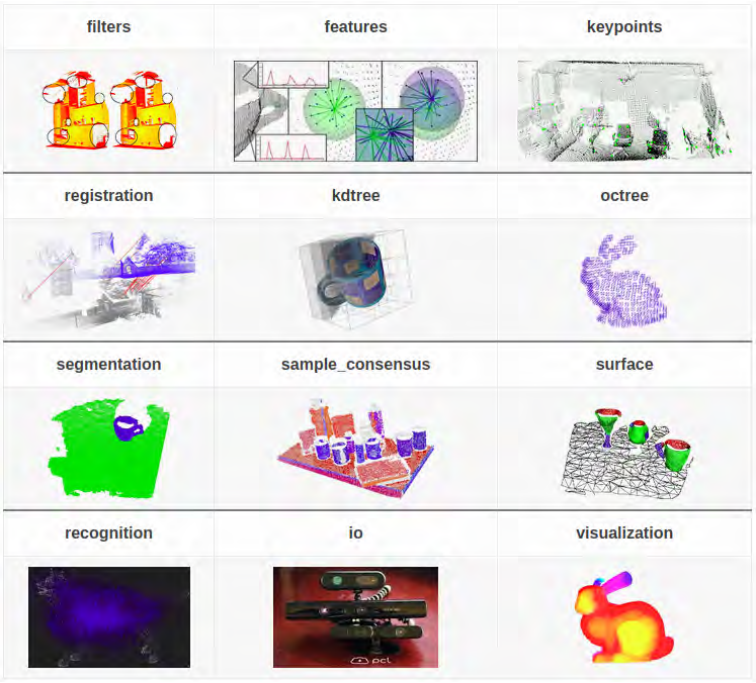


Figura 4.2: Diferentes opciones que ofrece la librería PCL, Rusu y Cousins (2011)

En concreto el paquete empleado para trabajar con PCL y ROS ha sido el creado por Sato (2022), el cual contiene implementados códigos en C++ con diferentes algoritmos de PCL para el procesamiento y transformaciones.

4.4 PCD

PCD es un formato de guardado de archivos utilizado por PCL. Contiene un encabezado que identifica y declara propiedades de los datos de la nube de puntos almacenados en el archivo, LLC (1999). Las ventajas que ofrece PCD frente a otros formatos de archivos son la habilidad de almacenar y procesar conjuntos de nubes de puntos ordenados, uso de datos binarios los cuales son los más rápidos de cargar o almacenar en el disco, flexibilidad de almacenamiento de datos de diferentes tipos y capacidad de obtener histogramas multidimensionales para descriptores de características. El encabezado se divide en los siguientes apartados:

- **VERSION**: se especifica la versión del archivo. La última versión actualmente es la 7 (PCD_V7).
- **FIELDS**: se especifica el nombre de cada campo que puede tener cada punto en la nube. Si únicamente se van a representar las posiciones tridimensionales, se definirá como *FIELDS x y z*, si además también se quieren representar los puntos con colores se definirá como *FIELDS x y z rgb*. Además de las formas mencionadas anteriormente hay más.
- **SIZE**: se especifica el tamaño de cada campo de los *FIELDS* en bytes. Si quisiéramos representar 4 campos con variables de tipo *int* se definiría como *SIZE 4 4 4 4*. Las diferentes opciones se listan a continuación:
  - unsigned char / char se representan con 1 byte
  - unsigned short / short se representan con 2 bytes
  - unsigned int / int / float se representan con 4 bytes
  - double se representa con 8 bytes
- **TYPE**: define el tipo de cada campo con un carácter. Los posibles caracteres son los siguientes:
  - **I**: representa los tipos *signed types* como int8 (char), int16 (short) y int32 (int).
  - **U**: representa los *unsigned types* como uint8 (unsigned char), uint16 (usigned short) y uint32 (unsigned int).
  - **F**: representa los tipos *float*
- **COUNT**: define el número de elementos que hay en cada campo individualmente.
- **WIDTH**: representa el número de puntos que hay en una fila. Esto es útil si la nube está definida como una matriz en la que hay puntos repartidos formando filas y columnas. Para mayor simplicidad se puede considerar que la nube está definida como un vector de tamaño (*WIDTH*,1) en el que cada elemento se corresponde con la información de cada punto de la nube, con lo que *WIDTH* se correspondería con el número total de puntos de la nube.

- **HEIGHT**: En caso de haber considerado que la nube está contenida en un vector de tamaño (*WIDTH*,1), la segunda dimensión del vector se corresponde con *HEIGHT*, el cual tiene un valor de 1.

- **VIEWPOINT**: determina el punto de vista desde el que se capturaron los puntos. Se expresa como una traslación (*tx ty tz*) junto con los cuaterniones (*qw qx qy qz*)

- **POINTS**: representa el número total de puntos en la nube

- **DATA**: contiene los valores de cada punto en la nube

```
# .PCD v.7 - Point Cloud Data file format
VERSION .7
FIELDS x y z rgb
SIZE 4 4 4 4
TYPE F F F F
COUNT 1 1 1 1
WIDTH 213
HEIGHT 1
VIEWPOINT 0 0 0 1 0 0 0
POINTS 213
DATA ascii
0.93773 0.33763 0 4.2108e+06
0.90805 0.35641 0 4.2108e+06
0.81915 0.32 0 4.2108e+06
...
```

(a) Código formato PCD

(b) Visualización 3D

Figura 4.3: Captura de un fragmento del código de almacenamiento de una nube de puntos en formato PCD y su visualización 3D,LLC (1999)

4.5 OpenCV

Es una biblioteca de software de aprendizaje automático y visión artificial de código abierto. La biblioteca tiene más de 2500 algoritmos optimizados de última generación, OpenCV (2022).

4.6 ROS

Ros es un metasistema operativo de código abierto para ayudar a desarrolladores en la creación de aplicaciones robóticas.Proporciona abstracción de hardware, controladores de dispositivos, bibliotecas, visualizadores, paso de mensajes, administración de paquetes, etc, *ROS/ Introduction* (2018). Además ROS es compatible con multitud de bibliotecas de movilidad, manipulación y percepción. ROS funciona con un sistema de publicaciones/subscripciones en *topics*.

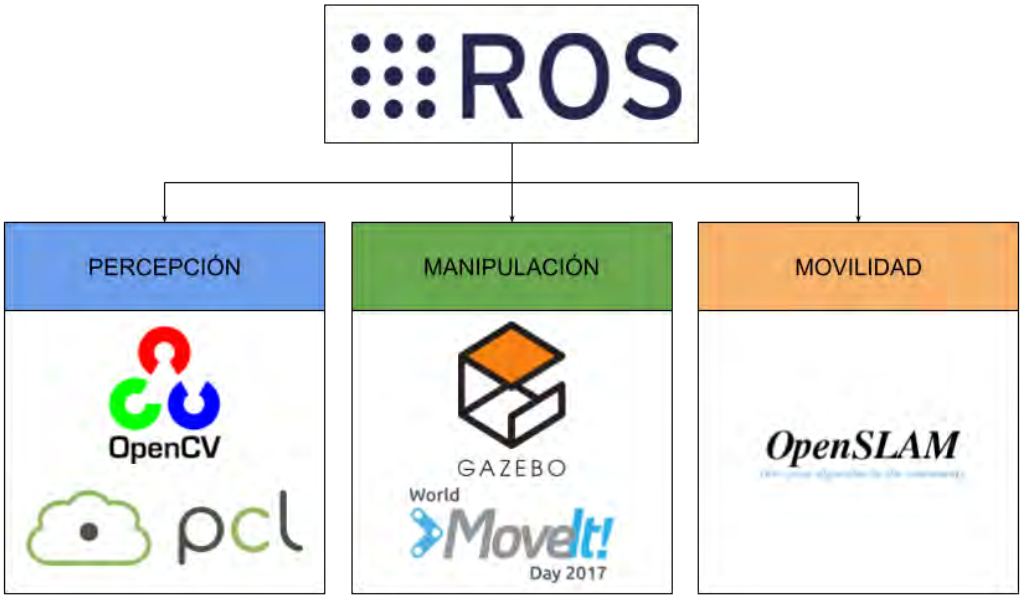


Figura 4.4: Representación de algunas de las librerías compatibles con ROS

4.7 RosBag

Consiste en un paquete que incorpora múltiples herramientas para grabar y reproducir *topics* en ROS. Las herramientas que proporciona permite trabar con *bags*, *RosBag* (2020). Las *bags* son un formato de archivo en ROS para almacenar datos de mensajes. Se han desarrollado multitud de herramientas para almacenarlas, procesarlas, analizarlas y visualizarlas, *Bags* (2022).

4.8 URDF

Se trata de un archivo formato XML para representar modelos de robots. Representan la apariencia del robot y la acción prevista. Los URDF se dividen en *links* (eslabones) y en *joints* (articulaciones).

Link	Definition
collision	Set information for link collision counting
visual	Set visualization information for links
origin	Set visualization information for links
mass	Weight of link [kg] set
inertial	Set inertia information for the link
geometry	Enter the shape of the model (Box, cylinder, sphere)

(a) Opciones para eslabones

Joint	Definition
parent	Joint Parent Link
child	Joint Child Link
origin	Convert parent link coordinate system to child link coordinate system
axis	Rotation axis setting
limit	Set joint speed, force, and radius

(b) Opciones para articulaciones

Figura 4.5: Opciones en los archivos URDF para describir los eslabones (*links*) y articulaciones (*joints*),Kang y cols. (2019)

Dentro de los URDF podemos encontrar diferentes archivos que simplifican y aumentna la capacidad de representar el modelo del robot. En concreto se van a comentar los archivo: *XACRO*, *TRANS* y *STL*

- **.XACRO**: es un lenguaje de macros para XML que se emplea para simplificar la creación de URDFs, *Using Xacro to Clean Up a URDF File* (2021).
- **.TRANS**: es una extensión de los modelos URDFs para describir la relación entre un actuador y una articulación mediante elementos de transmisión, *URDF Transmissions* (2017).
- **.STL**: es un formato de mallas (*meshes*) que representan eslabones con geometrías complejas a partir de triángulos sólidos, Béchet y cols. (2002).

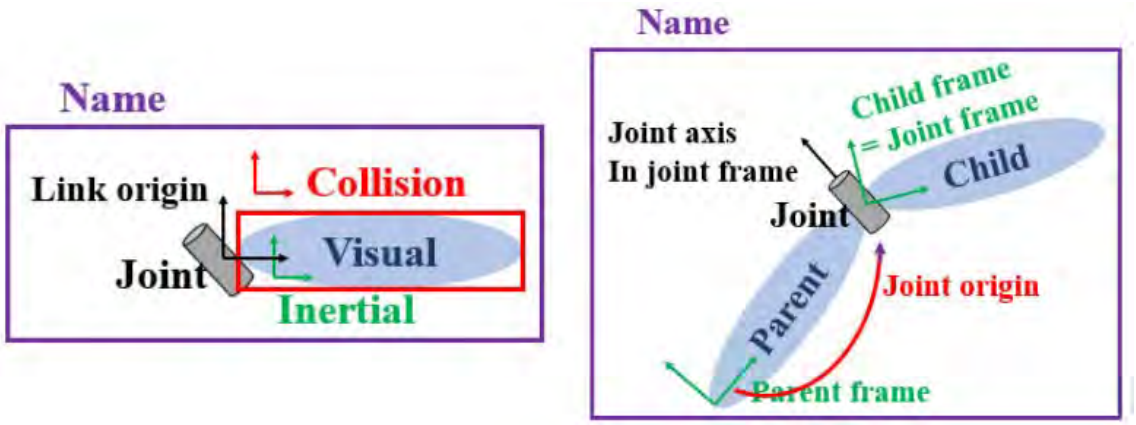


Figura 4.6: Representación de la estructura de los eslabones (*links*) y las articulaciones (*joints*), Kang y cols. (2019)

4.9 RVIZ

Rviz es una herramienta de de visualización 3D para ROS que proporciona interfaces extensibles tanto para visualización como para control de robots, Hart y cols. (2015). Entre otras cosas, Rviz permite cargar modelos URDF de robots y controlar sus articulaciones.

4.10 GAZEBO

Gazebo es un simulador 3D multi-robot con dinámicas. Ofrece la posibilidad de simular con precisión y eficiencia multitud 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. Por tanto Gazebo es un motor de físicas robusto y gráficos de alta calidad, "Ros-gazebo. una valiosa Herramienta de Vanguardia para el Desarrollo de la Robótica" (s.f.). Gazebo es una de las herramientas disponibles en ROS.

4.11 TOWR

TOWR es un *framework* desarrollado en C++ basado en Eigen para la optimización de trayectorias de robots con patas. TOWR proporciona secuencias de marcha, tiempos de paso, puntos de apoyo y movimientos del cuerpo en 6D en diferentes entornos para resolver problemas de locomoción con piernas, A. Winkler (2021). El simulador dispone de multitud de parámetros que se pueden ajustar para obtener la trayectoria deseada para un robot con patas. La visialización se realiza en Rviz. En la siguiente figura (Figura 4.7) se puede apreciar el terminal con algunas de las configuraciones de la trayectoria que se quiere realizar y de su visualización.

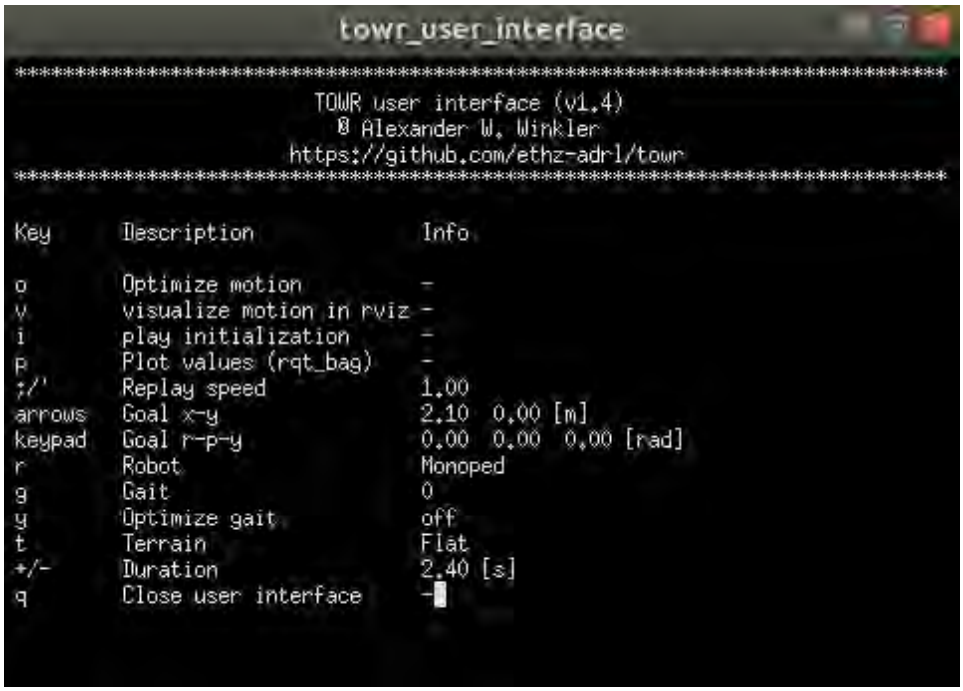


Figura 4.7: Terminal de configuración de TOWR

Los parámetros que se pueden ajustar en el terminal de la Figura 4.7 son los siguientes:

- **Optimiza motion**: permite inicializar la optimización de la trayectoria en función de los parámetros de entrada como la posición inicial, la posición final deseada y los parámetros ajustados en el terminal.
- **visualize motion in rviz**: visualiza el movimiento del robot
- **Play initialization**: inicializa la visualización del movimiento
- **Plot values (qrt_bag)**: imprime los valores almacenados en un *bag*
- **Replay speed**: permite modificar la velocidad a la que se visualiza el movimiento
- **Goal x-y**: define la posición final

- **Goal r-p-y**: define la orientación final
- **Robot**: permite elegir entre diferentes modelos de robots
- **Gait**: define el número de pasos para completar la trayectoria
- **Optimize gait**: optimiza el número de pasos
- **Terrain**: permite seleccionar entre diferentes terrenos
- **Duration**: establece la duración total de la trayectoria
- **Close user interface**: cierra el terminal de configuración

En la siguiente figura se puede ver una captura del entorno de TOWR con un robot bípedo de ejemplo, un terreno plano y una posición final deseada por defecto.

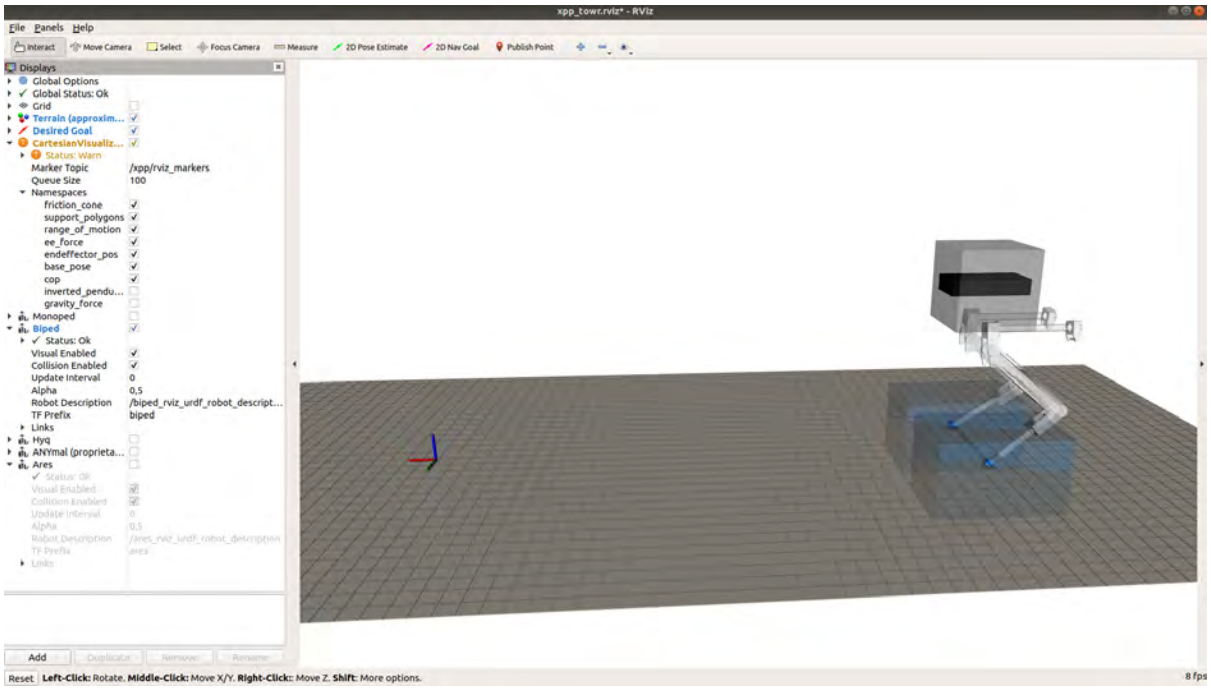


Figura 4.8: Captura del entorno de simulación TOWR

4.12 Xpp

Xpp es un paquete de visualización de planes de movimiento para robots con patas. Dibuja áreas de apoyo, fuerzas de contacto y trayectorias de movimiento en Rviz, A. W. Winkler (2017). Este paquete se puede emplear para visualizar las trayectorias generadas por TOWR.

4.13 Pinocchio

Pinocchio es una biblioteca de C++ que contiene algoritmos de cuerpo rígido de última generación para sistemas poliarticulados basados en los algoritmos revisados de Roy Feartherstone, Featherstone (2014). Entre sus principales características se puede encontrar el cálculo de cinemática directa, dinámica directa/inversa, dinámica centroidal, Carpentier y cols. (2015–2021).

4.14 PlotJuggler

PlotJuggler es una herramienta que permite la visualización de gráficas de manera rápida, potente e intuitiva, Faconti (2016). Entre otras funciones, permite visualizar de archivos *bag* procedentes de *topics* de ROS en forma de gráficas.

4.15 Realsense-ros-2.3.2

Consiste en un paquete desarrollado por Doronhi (2021), para emplear las cámaras RealSense junto con ROS, permitiendo inicializarlas, visualizar las imágenes 2D y 3D captadas por la cámara, e incluye multitud de códigos de ejemplo para desarrollar nuevos códigos. Este paquete está disponible en las versiones Kinetic, Melodic y Noetic.

5 Desarrollo

Es este apartado se van a detallar las fases en las que se ha dividido el trabajo y las tareas desempeñadas en cada una de ellas. En la figura 4.1, mostrada anteriormente, se puede ver una visión general del trabajo.

5.1 Control de la Cámara

Para realizar las capturas de las nubes de puntos se empleó una cámara Intel REALSENSE LiDAR L515 (Figura 5.1.A). Esta cámara de profundidad con tecnología LiDAR permite capturar tanto imágenes bidimensionales RGB como imágenes tridimensionales a una distancia comprendida entre 0.25 y 9 metros, con una precisión de entre 5 y 14 milímetros.

(a) Cámara RealSense L515 LiDAR

(b) Imagen RGB 2D real

(c) Imagen 3D capturada

Figura 5.1: Cámara de profundidad empleada y comparación de la imagen 2D y 3D de una escena capturada, *Intel REALSENSE LiDAR Camera L515* (s.f.)

Como se ha comentado anteriormente en los objetivos (Apartado 2), se pretende representar lo captado por la cámara en el simulador TOWR, el cual funciona con ROS, por tanto se debe buscar la forma de compatibilizar la cámara con ROS.

5.1.1 Inicialización de la cámara

El primer paso para obtener nube de puntos consiste en iniciar la publicación de la información captada por la cámara. Para ello se ha empleado un paquete de ROS desarrollado por Doronhi (2021) en el que al ejecutar uno de los códigos que contiene, se inicializa todo lo necesario para que se creen una serie de *topics*, en los que se publica la correspondiente información sobre la imagen RGB bidimensional, la nube de puntos tridimensional y la orientación de la cámara entre otros. Con este paquete se consigue poder visualizar en Rviz la imagen tridimensional de la escena que está captando la cámara (Figura 5.1.C). Cabe destacar que la nube de puntos generada se encuentra en formato *PointCloud2*.

5.1.2 Creación de un entorno de pruebas

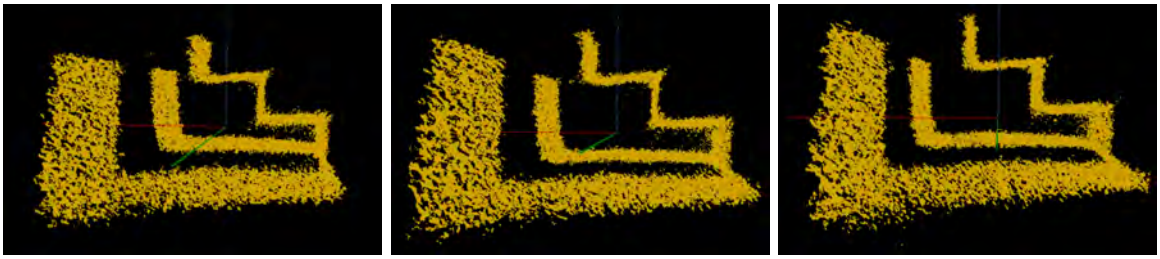
Con el fin de realizar pruebas del control de la cámara, captura de la escena y procesamiento de las nubes de puntos, se ha creado un entorno de experimentación compuesto de una escalera de cartón, fabricada cumpliendo íntegramente las especificaciones estipuladas en el Real Decreto 314/2006 del 17 de marzo, en el que se establece las dimensiones de una escalera de uso general, BOE (2006). Una vez realizadas las pruebas e implementados los códigos necesarios, se ha procedido a efectuar experimentos con escaleras reales.

(a) Escalera de lado **(b)** Escalera de frente **(c)** Dimensiones escalera

Figura 5.2: Imágenes de la escalera de cartón creada para realizar pruebas iniciales. En la Figura C, se puede ver una escuadra para confirmar que los escalones están a 90º. Las dimensiones de la escalera son X:30, Y:40 y Z:20 centímetros.

5.1.3 Captura de las nubes de puntos

El siguiente paso es guardar una o varias copias de la escena que se está visualizando. Como se comentó en el trabajo Woo y cols. (2019), para una misma escena, debido al error producido por la propia cámara (entre otros), a cada instante de tiempo se generan nubes de puntos diferentes (*frames*). Para reducir el error de la nube, para cada escena, se capturaron un número de *frames* que se fue variando para comparar los resultados.



(a) Frame 1 **(b)** Frame 2 **(c)** Frame 3

Figura 5.3: Comparación de los diferentes *frames* capturados de una misma escena. Aunque son aparentemente idénticas, la posición de los puntos tiene pequeñas variaciones entre *frames*.

Para capturar cada *frame* se realiza un *rosbag record* el cual genera un archivo *.bag* en el que se graba toda la información publicada en los *topics* especificados durante un período predefinido. Mediante una herramienta desarrollada por Aposhian (2017), que permite transformar un archivo *.bag* en uno PCD, se almacena la nube de puntos en formato PCD. Por otro lado, para obtener el ángulo de la cámara, se realiza la reproducción del archivo *.bag* almacenado empleando la herramienta *rosbag play*, haciendo que los datos almacenados se vuelvan a publicar en los *topics* durante el mismo tiempo con el que se grabaron. Durante esta reproducción, mediante las herramientas que proporciona ROS, se almacena la posición angular publicada en uno de los *topics* en un archivo de texto que almacena la información en "bruto". Seguidamente se ejecuta un código que accede al archivo de texto en bruto con la posición angular de la cámara, extrae únicamente el valor de inclinación de la cámara y lo guarda en otro archivo de texto junto con el nombre de la escena, el del *frame* y la altura de la cámara al realizar la captura de la escena. Este último archivo de texto realiza la función de registro de capturas, dado que al capturar cada frame, el archivo de texto en bruto se sobrescribe, quedando únicamente la posición angular de la última captura.

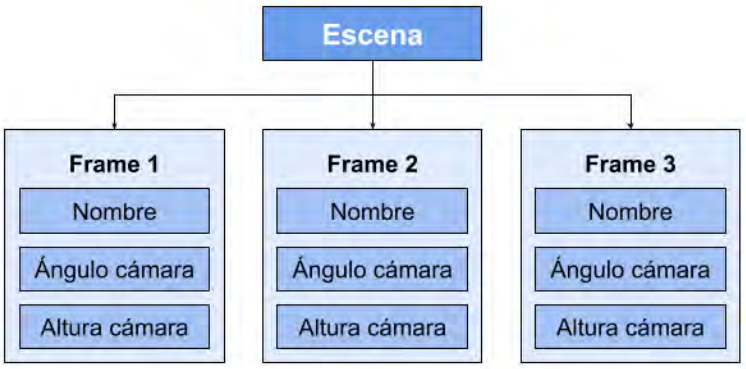


Figura 5.4: Esquema de la organización de los datos del registro de nubes.

5.2 Procesamiento de la nube de puntos

A continuación se van a explicar el los pasos del algoritmo implementados para representar la nube de puntos capturada, en el simulador TOWR. Como se ha comentado anteriormente, el trabajo se va a centrar en nubes de puntos de escaleras. El simulador TOWR únicamente requiere la altura y profundidad de los escalones para poder representarlo. Cabe destacar que no se considera la posibilidad de que no haya una escalera en la nube capturada y que siempre estará centrada en la escena.

5.2.1 Obtención de la nube de puntos capturada

El primer paso consiste en seleccionar la escena con la que se quiere trabajar para que se localice en el registro los nombres de todos los *frames* y los respectivos ángulos de inclinación de la cámara con los que fueron capturados. El algoritmo implementado obtendrá como resultado las profundidades y alturas de cada escalón de cada *frame* y las comparará con el resto *frames* para obtener un resultado general de la escena.

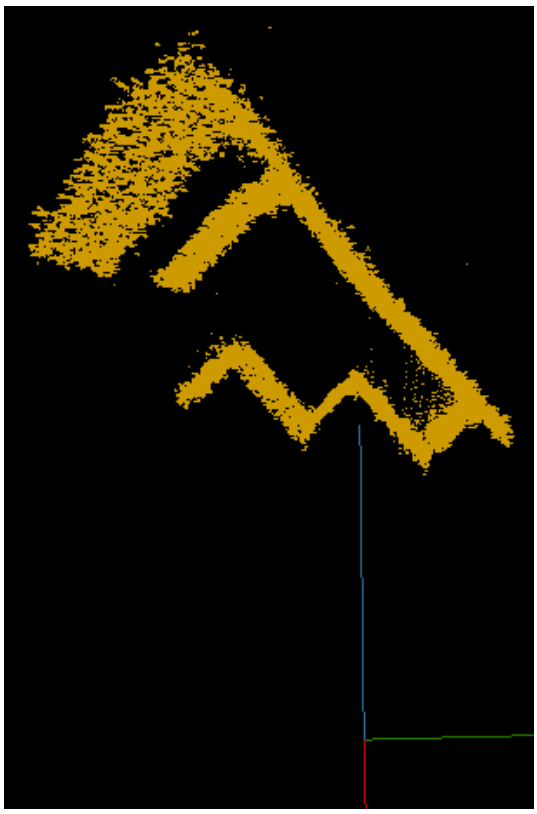


Figura 5.5: Nube originalmente capturada

5.2.2 Rotación

Con el objetivo de simplificar cálculos y haciendo uso del ángulo de inclinación de la cámara, se realiza una traslación y rotación de la posición original de la nube, centrándola y girándola para alinearla con los ejes principales.

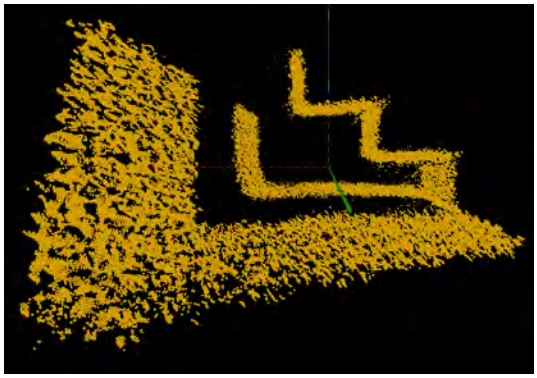


Figura 5.6: Nube de puntos trasladada y rotada

5.2.3 Sección

Seguidamente, se seleccionan los puntos que estén dentro de un rango de distancia en ambos sentidos del eje Y (Figura 5.7). De este modo, se eliminan elementos indeseados de los extremos de la nube como paredes laterales o barandillas. El tamaño de la sección se determina en función a una constante. Cuanto mayor sea la sección, mayor precisión se obtendrá en el procesamiento, pero aumentará el riesgo de que se incluyan algunos de los elementos indeseado anteriormente mencionados con lo que se producirían errores en el procesamiento.

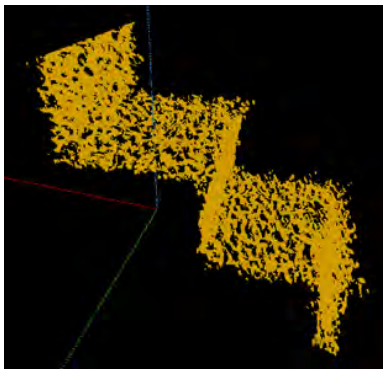
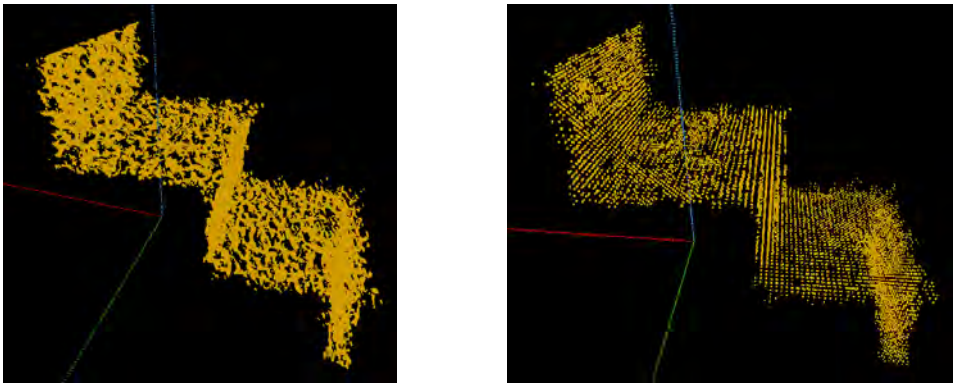


Figura 5.7: Sección de la nube seleccionada.

5.2.4 Redondeo de coordenadas

Una vez seccionada la escalera, las coordenadas de los puntos que la componen tienen más de 10 decimales, siendo por ejemplo x:-0.7308157682418823, y:-0.6227220296859741 y z:1.4152500629425049. En este paso se redondean las coordenadas a 2 decimales para simplificar y uniformizar la nube, de modo que ahora el puntos de ejemplo anterior quedan como x:-0.73, y:-0.62 y z:1.42. De la simplificación realizada en este paso se beneficiaran los siguientes.



(a) Nube sin redondeo **(b)** Nube con redondeo

Figura 5.8: Comparación de la nube antes y después de realizar el redondeo de las coordenadas de los puntos.

5.2.5 Compresión en un perfil

Tras redondear las coordenadas de los puntos, se procede a poner a ”0” las componentes ’Y’ de todos los puntos, eliminando una dimensión, comprimiendo todos los puntos en un mismo plano formando un perfil de la escalera.

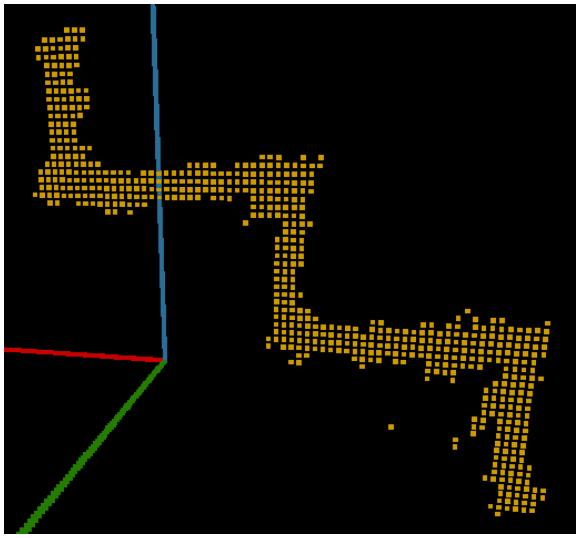


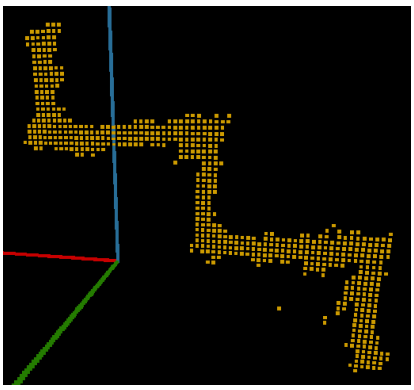
Figura 5.9: Perfil de la escalera tras comprimir los puntos en un único plano

5.2.6 Eliminación de los puntos repetidos

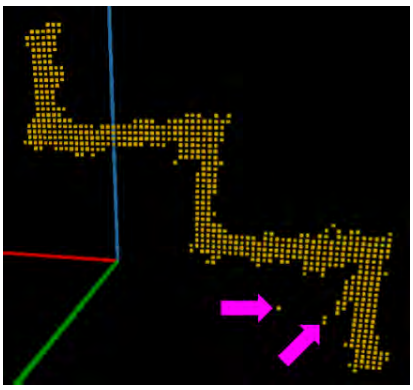
Debido al redondeo realizado en el apartado 5.2.4 y a la compresión realizada en el apartado 5.2.5, se ha favorecido la posibilidad de que para una misma posición hayan varios puntos con las mismas coordenadas, visualizándose como un único punto, por lo que visualmente es idéntica a la Figura 5.9. Con el objetivo de simplificar más la nube, se registra el número de veces que se repite cada punto y se eliminan los puntos repetidos dejando exclusivamente uno en cada posición.

5.2.7 Eliminación de los puntos aislados

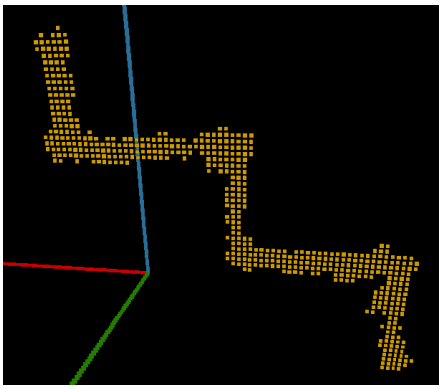
Se consideran puntos aislados a los que están a una distancia mayor que un umbral de sus puntos vecinos o tienen un número de vecinos menor que un umbral. Para determinar si un punto está aislado se ha de comparar con otros puntos, es por ello que se realiza este paso en este instante y no antes, ya que en cada apartado se van reduciendo el número de puntos de la nube y se reduce el número de comparaciones que se deben realizar.



(a) Perfil original



(b) Perfil con algunos puntos aislados señalados

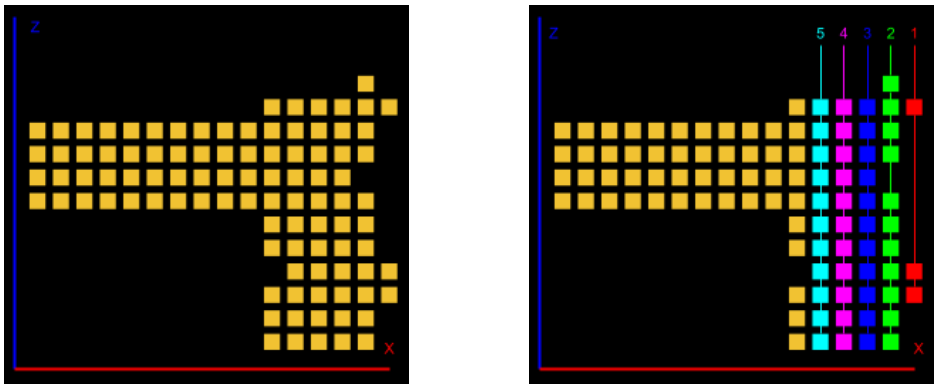


(c) Perfil sin puntos aislados

Figura 5.10: Comparación del perfil de la escalera con y sin los puntos aislados

5.2.8 Extracción de los puntos más repetidos

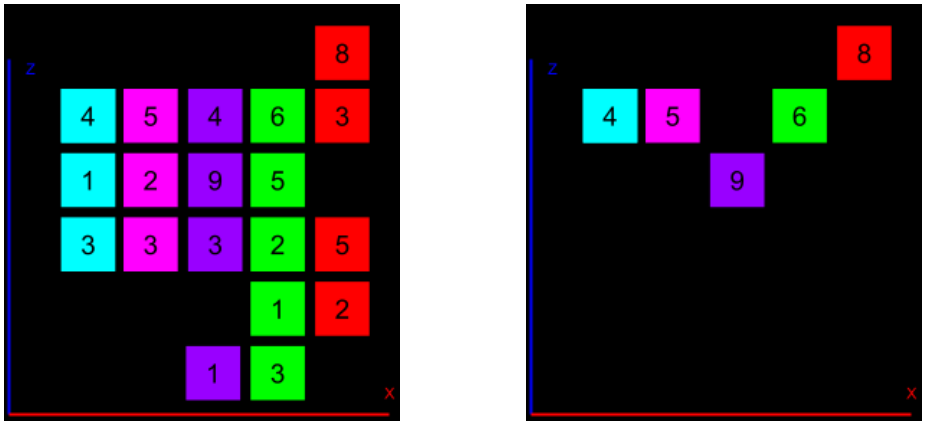
Como se puede ver en la Figura 5.9 del perfil de la nube, al redondear los puntos, quedan situados en forma de cuadrícula uniforme. A continuación agrupan los puntos en rebanadas según su distancia sobre el eje 'X'.



(a) Ejemplo puntos **(b)** Agrupación de los puntos

Figura 5.11: Representación de la agrupación de los puntos según su distancia sobre el eje 'X'. Cada grupo de puntos coloreados distinto del amarillo representa una rebanada.

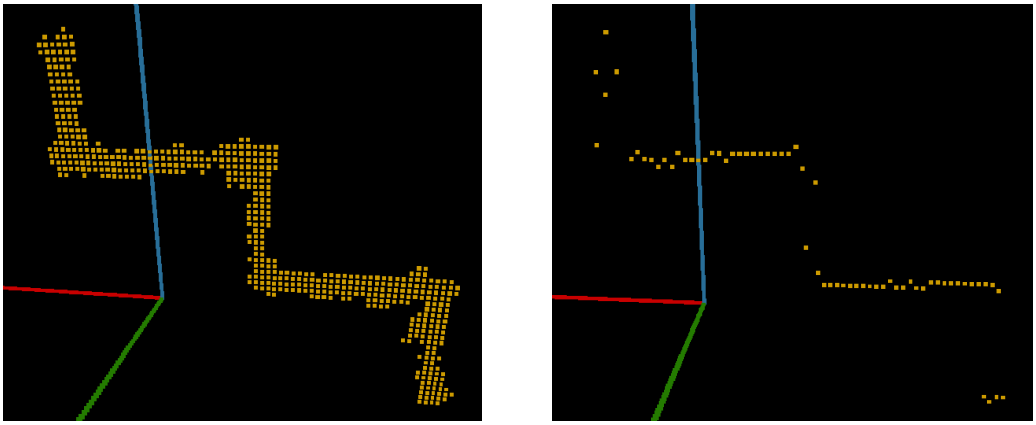
Como se comentó en el apartado 5.2.6, se registraron los puntos repetidos y se dejó únicamente uno por cada posición. En este apartado se realiza la consideración de que que cuantos más puntos repetidos hayan en una posición, más fiable será ese punto respecto de sus vecinos del grupo horizontal. Siguiendo esta consideración, de cada grupo se extrae el punto con mayor número de repeticiones, quedando éste como único miembro del grupo. En la siguiente figura se puede ver un ejemplo de lo anteriormente comentado.



(a) Ejemplo repeticiones puntos **(b)** Extracción de los más repetidos

Figura 5.12: Representación de la extracción de los puntos más repetidos de cada rebanada. El número dentro de cada puntos simboliza el número de veces que se repite.

En la siguiente figura se muestra un ejemplo de la selección de los puntos con mayor número de repeticiones con una nube real.

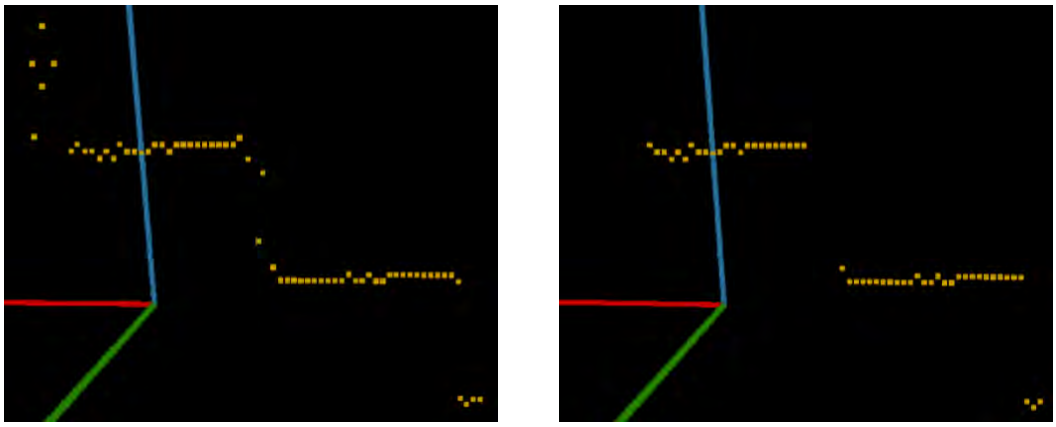


(a) Nube con todos los puntos **(b)** Nube con los puntos más repetidos

Figura 5.13: Comparación entre la nube de puntos con todos los puntos (A) y la nube únicamente con los más repetidos (B)

5.2.9 Eliminación de los puntos intermedios

Una vez extraídos los puntos que más se repiten, se procede a eliminar los puntos intermedios. Se consideran puntos intermedios los puntos que no están alineados con otros horizontalmente, es decir, que no forman superficies horizontales que se corresponderían con la huella de los escalones. La eliminación de puntos intermedios es una variación de la eliminación de los puntos aislados (Apartado 5.2.7), de modo semejante se eliminan los puntos que no tengan vecinos dentro de un umbral horizontal.



(a) Nube con puntos intermedios **(b)** Nube sin puntos intermedios

Figura 5.14: Comparación entre una nube con y sin puntos intermedios.

5.2.10 Obtención de la altura y la profundidad del escalón

En este apartado primero se extraen los puntos extremos de la nube, los cuales se corresponderán con el inicio y final de cada escalón de la escalera. Estos puntos extremos son los que definen las dimensiones de una escalera (altura y profundidad).

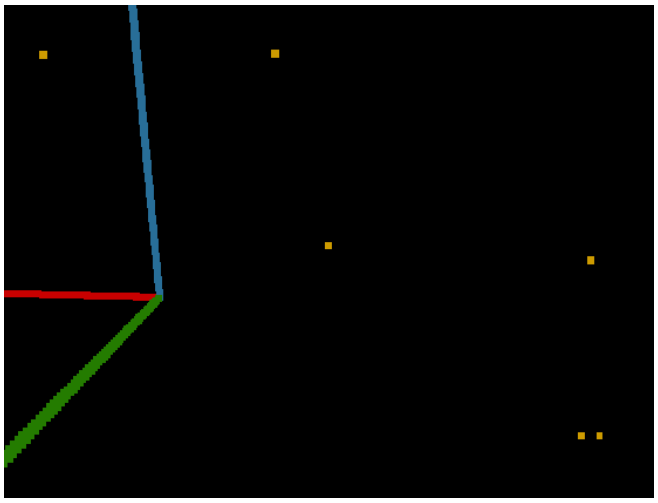
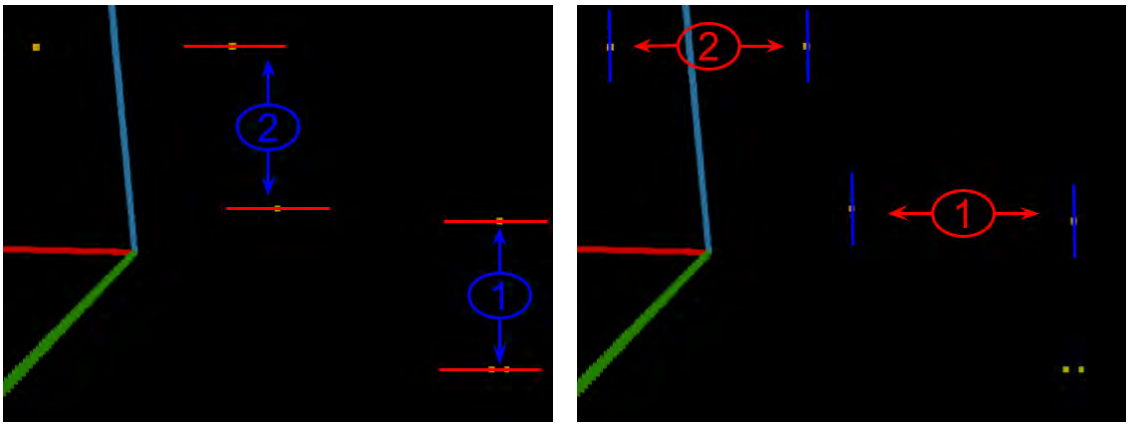


Figura 5.15: Comparación entre una nube con y sin puntos intermedios.

Para extraer las alturas y profundidades de los escalones, se restan las posiciones de los puntos extremos. En la Figura 5.16 se puede ver representadas las zonas de las que se toman las medidas.



(a) Obtención de la altura **(b)** Obtención de la profundidad

Figura 5.16: Representación de la obtención de la altura (A) y la profundidad (B) de los escalones.

5.2.11 Filtrado y cálculo de los valores

Tras obtener las alturas y profundidades de los escalones, se realiza una evaluación de los valores. Inicialmente se descartan los valores de altura de escalones que estén fuera del rango de 0.13-0.22 metros, este rango se determina a partir de la legislación sobre dimensiones de escalones en escaleras de uso general junto con un cierto margen de holgura. Seguidamente se realiza el mismo procedimiento con las profundidades con un rango de 0.25-0.35 metros. Por último, para las alturas y las profundidades de cada *frame*, se calcula la moda, dado que se pretende seleccionar el valor de altura y profundidad que más se repite. De este modo almacena un único valor de altura y otro de profundidad de cada *frame*.Una vez realizado el procedimiento con todos los *frames*, se vuelve a realizar la moda de las alturas y las profundidades, obteniendo un único valor final de altura y otro de profundidad para toda la escalera. Se obtienen únicamente estos dos valores ya que se considera que la escalera es uniforme.

5.2.12 Reconstrucción de la escalera

Una vez obtenidos los valores de altura y profundidad de los escalones, se cuenta el número de valores válidos (dentro del rango) que obtuvieron en el aparatado anterior antes de hacer la moda de los valores, de este modo para cada *frame* se obtiene un número de escalones. Seguidamente se realiza nuevamente la moda del número de escalones detectado en cada *frame*. Sabiendo el número de escalones y las dimensiones se reconstruye los valores de la escalera escalón a escalón.

5.2.13 Escalado

Debido a que tras realizar una serie de pruebas, se llegó a la conclusión de que el exoesqueleto no fue diseñado con las dimensiones óptimas para subir escaleras reales ya que era demasiado pequeño, se ha decidido añadir un escalado para reducir uniformemente el tamaño de la escalera y hacerla proporcional al exoesqueleto. De este modo se equiparan todas las dimensiones y se obtienen resultados más exactos. El escalado que se ha realizado de la escalera es de **7/9**.

5.2.14 Traspaso de datos a TOWR

Por último, se transfieren los resultados al simulador TOWR. Para ello se recopia el archivo que genera los terrenos sustituyendo con los valores obtenidos. El simulador genera el terreno por defecto a partir del perfil en 2D (X,Z) y extruye la dimensión que falta a lo ancho de todo el terreno (Y). En la siguiente figura se puede ver una representación del modo en el que TOWR genera un terreno por defecto con escaleras.

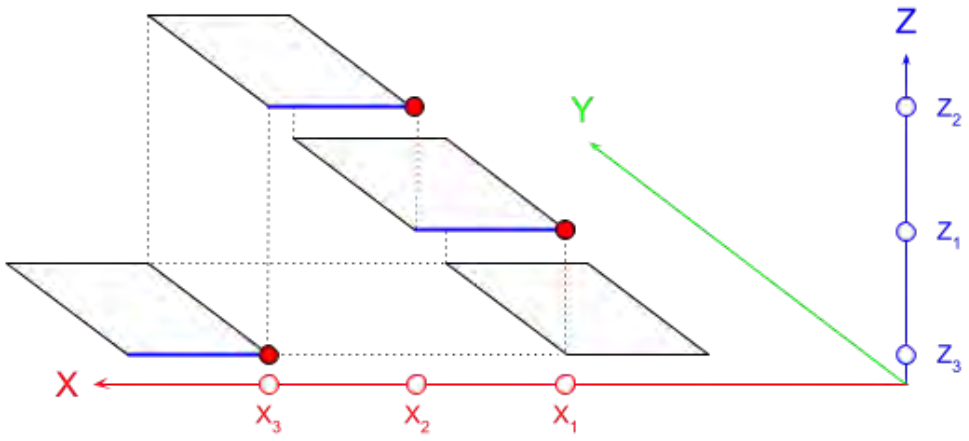


Figura 5.17: Representación de la generación de terrenos de escaleras en TOWR. Los puntos rojos indican las zonas en las que el terreno se modifica.

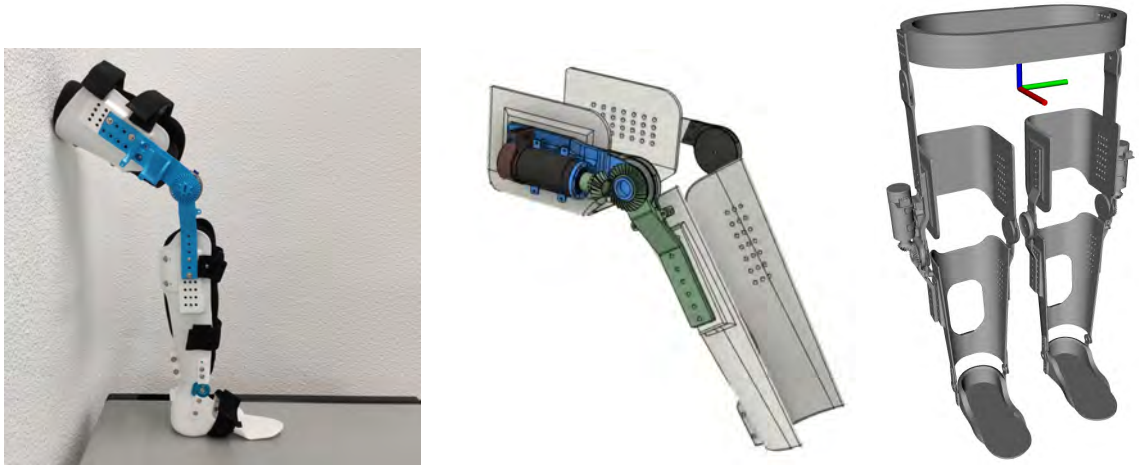
El método por defecto que emplea TOWR para modificar el terreno es mediante intervalos. En la siguiente expresión se puede ver un ejemplo de dicho intervalo.

$$\begin{Bmatrix} x > x_1 \rightarrow z = z_1 \\ x > x_2 \rightarrow z = z_2 \\ x > x_3 \rightarrow z = z_3 \\ \dots \end{Bmatrix} \tag{5.1}$$

Como se ha podido comprobar, durante el procesamiento de la nube no se ha empleado ningún algoritmo "estándar" para la reducción del número de puntos y del tiempo de procesamiento como los filtros de VoxelGrid, aunque sí se ha ido reduciendo el número de puntos en cada apartado. Esto se debe a que el tiempo de procesamiento no se ha considerado completamente relevante, mientras se trate de unos segundos, ya que TOWR requiere de un cierto tiempo para generar las trayectorias y por tanto no se puede ni se pretende trabajar en tiempo real.

5.3 Modelo 3D (URDF)

En este apartado se van a describir las características del URDF del exoesqueleto de miembro inferior empleado. El modelo está diseñado a partir de una órtesis real de rodilla (con sujeciones para la pierna y el muslo) y un motor para accionarla. El diseño original se desarrolló dentro del proyecto ARES de la iniciativa Exoforge del grupo de investigación HURO, ganadores de los Premios Impulso 2022 en la categoría de Accesibilidad, cuyo objetivo es el desarrollar un exoesqueleto de rodilla de bajo coste. A partir del diseño realizado durante el proyecto, se solicitó al equipo de diseño una nueva versión para poder emplearla en este trabajo (Figura 5.18.).



(a) Órtesis real **(b)** Diseño de la órtesis **(c)** Exoesqueleto Ares

Figura 5.18: Diseño 3D realizado a partir de la órtesis en el proyecto ARES.

5.3.1 Descripción del modelo 3D

El modelo del exoesqueleto de miembro inferior tiene una altura de 0.851742 metros y una anchura de 0.415369 metros. Consta de 6 articulaciones rotacionales con la siguiente configuración

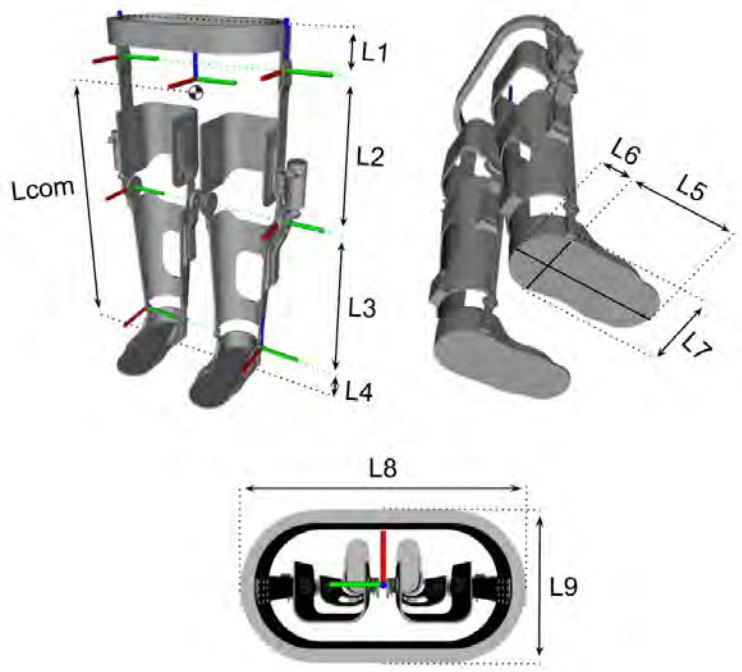


Figura 5.19: Representación del URDF del exoesqueleto de miembro inferior Ares. Lcom: distancia del suelo al centro de masas.

L	Tamaño (m)
L1	0.092
L2	0.335917
L3	0.35995
L4	0.063875
L5	0.194471887

L	Tamaño (m)
L6	0.054994483
L7	0.114653165
L8	0.415369
L9	0.22
Lcom	0.72566962

Cuadro 5.1: Valor de las medidas representadas en la Figura 5.19

5.4 Simulador TOWR

A continuación se va a explicar todo lo realizado en TOWR. Para poder visualizar el movimiento del robot, ha sido necesario introducirle al simulador el nuevo URDF de Ares y el paquete de Xpp. El espacio de trabajo empleado se divide en 3 apartados:

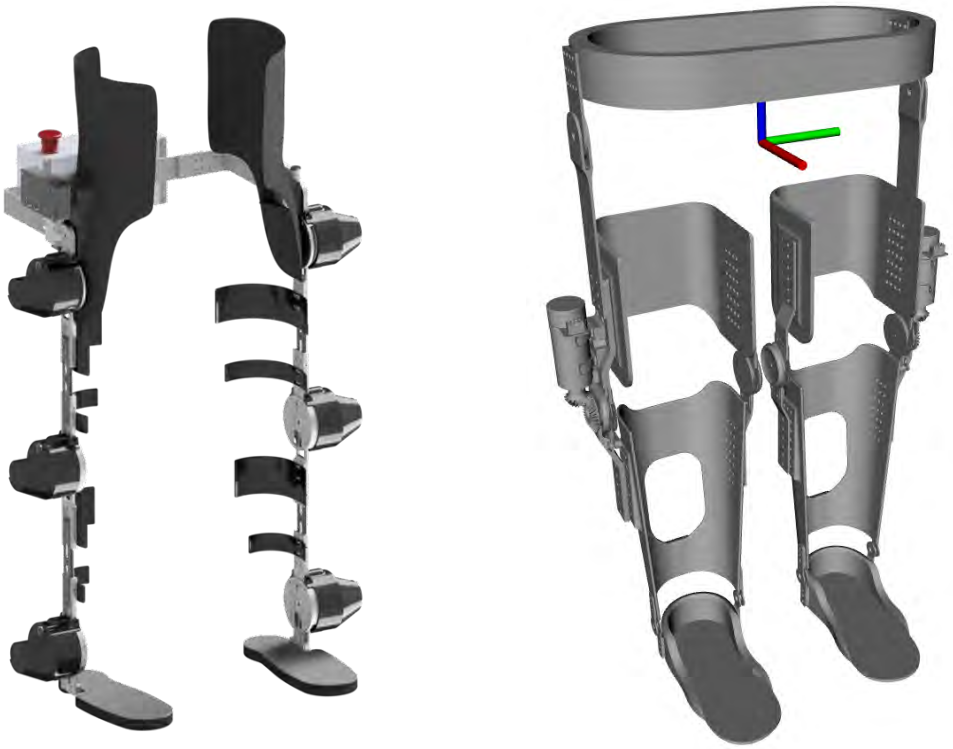
- **towr**: se encarga de generar las trayectorias en base a la información proporcionada y al ajuste de ciertos parámetros.

- **urdf**: contiene el URDF del exoesqueleto de miembro inferior Ares.

- **xpp**: realiza la visualización del exoesqueleto, reconstruyendo la información a partir de la posición que le proporciona TOWR y las dimensiones del URDF.

5.4.1 Incorporación del URDF

Para incorporar el URDF a TOWR se han seguido las indicaciones de los paquetes de GitHub *towr_h3* y *xpp_h3* de Bernat (2022) en el que se especifican los archivos que se han de modificar dentro del paquete de TOWR y de Xpp. En el paquete de Bernat (2022) se introduce en TOWR el robot H3. Dicho robot tiene características semejantes al de Ares, la tarea principalmente consiste en sustituir los parámetros característicos del robot H3 por los de Ares.



(a) Modelo 3D del Exo-H3 **(b)** Modelo 3D de Ares

Figura 5.20: Comparación de los modelos 3D de los exoesqueletos de miembro inferior H3 y Ares.

5.4.2 XPP

Al igual que para TOWR, en el paquete Xpp se han de modificar archivos y parámetros para realizar la visualización . Entre las principales modificaciones, se debe editar la cinemática inversa para reconstruir la visualización del URDF.

5.4.2.1 Cinemática inversa

Para la cinemática inversa se emplearon las mismas expresiones que las usadas en Bernat Iborra (2022) dado la similitud entre los exoesqueletos. En la imagen 5.21 se puede apreciar el esquema de la cinemática inversa y seguidamente se pueden ver las expresiones anteriormente mencionadas.

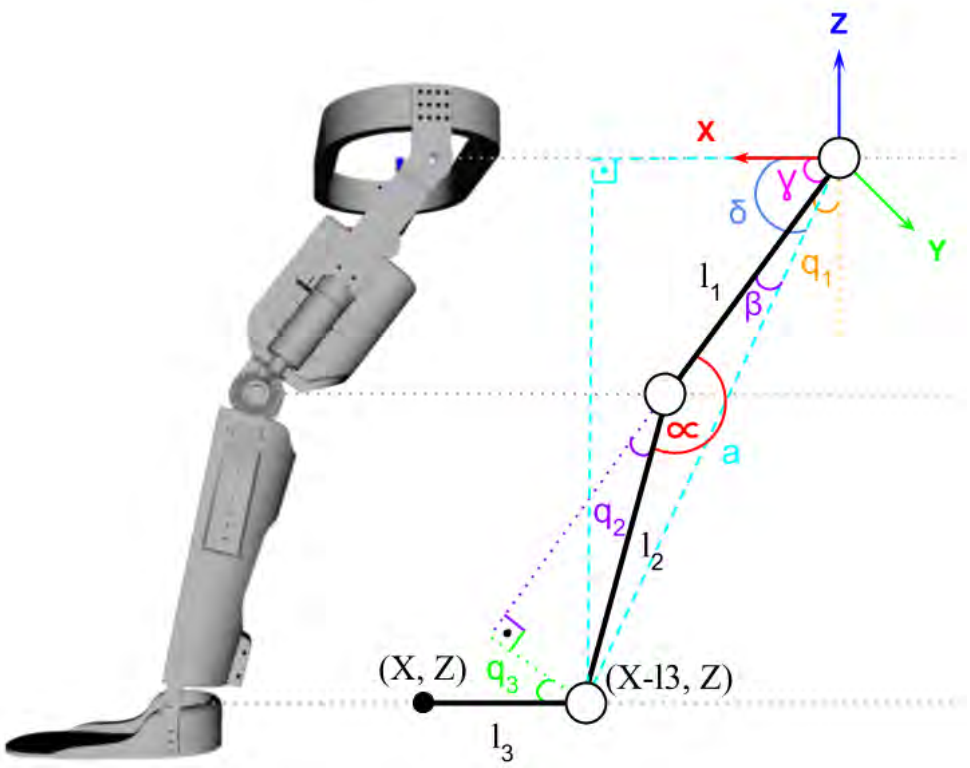


Figura 5.21: Esquema de la cinemática inversa del exoesqueleto Ares simplificado. En caso de colocar el efector final en la misma posición sobre el eje X que la articulación q_3, l_3 se consideraría 0.

- Cálculo de q_2:

$$q_2 + \alpha = \pi \rightarrow cos(q_2) = cos(\pi - \alpha) \rightarrow cos(q_2) = -cos(\alpha)$$

$$\begin{cases} a^2 = l_1^2 + l_2^2 - 2 \cdot l_1 \cdot l_2 \cdot cos(\alpha) \\ a^2 = (X - l_3)^2 + Z^2 \end{cases} \rightarrow cos(\alpha) = \frac{(X - l3)^2 + Z^2 - l_1^2 - l_2^2}{2 \cdot l_1 \cdot l_2 \cdot cos(\alpha)}$$

$$cos(q_2) = \frac{(X - l3)^2 + Z^2 - l_1^2 - l_2^2}{2 \cdot l_1 \cdot l_2 \cdot cos(\alpha)}$$

$$cos(q_2)^2 + sin(q_2)^2 = 1 \rightarrow sin(q_2)^2 = \sqrt{1 - cos(q_2)^2}$$

$$tan(q_2) = \frac{sin(q_2)}{cos(q_2)} \rightarrow q_2 = arctan\frac{\sqrt{1 - cos(q_2)^2}}{cos(q_2)} \tag{5.2}$$

- Cálculo de q_1:

$$\begin{cases} q_1 = \frac{\pi}{2} - \gamma \\ \gamma + \beta = \delta \end{cases} \rightarrow q_1 = -\delta + \beta$$

$$tan(\delta) = \frac{Z}{X - l_3} \rightarrow \delta = arctan\frac{Z}{X - l_3}$$

$$tan(\beta) = \frac{l_2 \cdot sin(q_2)}{l_1 + l_2 \cdot cos(q_2)} \rightarrow \beta = arctan\frac{l_2 \cdot sin(q_2)}{l_1 + l_2 \cdot cos(q_2)}$$

$$q_1 = \frac{\pi}{2} - arctan(\frac{Z}{X}) + arctan(\frac{l_2 \cdot sin(q_2)}{l_1 + l_2 \cdot cos(q_2)}) \tag{5.3}$$

- Cálculo de q_3:

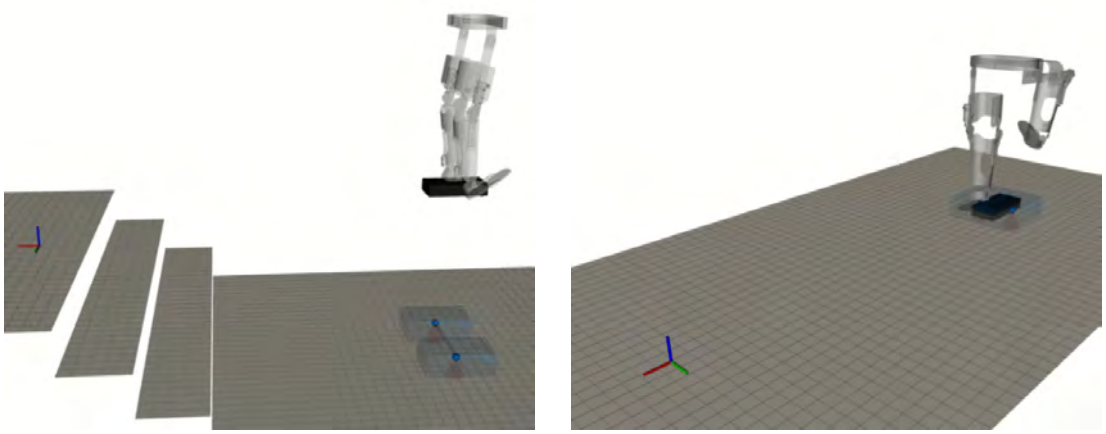
$$q_3 + \theta = \frac{\pi}{2} \rightarrow q_3 = \frac{\pi}{2} - \theta$$

$$tan(\theta) = \frac{Z - l_1 \cdot cos(q_2)}{X - l_3 - l_1 \cdot sin(q_2)} \rightarrow \theta = arctan\frac{Z - l_1 \cdot cos(q_2)}{X - l_3 - l_1 \cdot sin(q_2)}$$

$$q_3 = \frac{\pi}{2} - arctan(\frac{Z - l_1 \cdot cos(q_1)}{(X - l_3) - l_1 \cdot sin(q_1)}) \tag{5.4}$$

5.4.3 Ajustes realizados en TOWR

Tras realizar los pasos anteriores y al ejecutar el simulador TOWR, se puede visualizar el exoesqueleto en la posición inicial. Tras iniciar el cálculo de la trayectoria óptima e iniciar la marcha, se ha podido ver como no camina correctamente con los ajustes por defecto, por lo que es necesario realizar modificaciones. En la Figura 5.22.A el exoesqueleto está erróneamente elevado, el paralelepípedo negro que representa el centro de masas está situado en los pies y además se encuentra de espaldas al sentido de la marcha. En el Figura 5.22.B además de algunos de los errores de la Figura 5.22.A, tiene una de las piernas completamente contraída.



(a) URDF flotando en el aire **(b)** URDF con pierna contraída

Figura 5.22: Imágenes del exoesqueleto Ares al cargarlo en el simulador TOWR antes de realizar las modificaciones para que funcionase correctamente.

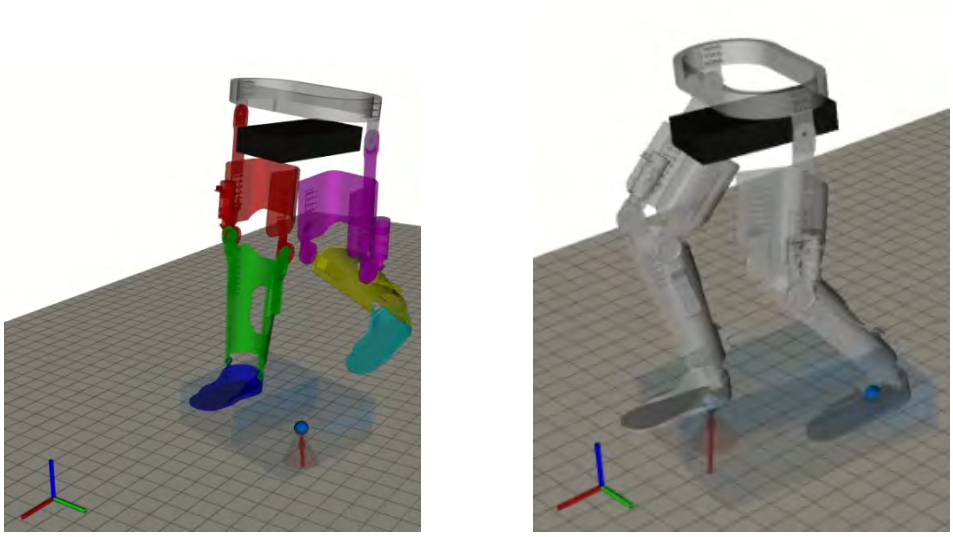
5.4.3.1 Cálculo de la matriz dinámica

Inicialmente se calculó la matriz dinámica para mejorar el movimiento del exoesqueleto. Para ello se hizo uso de la librería *pinocchio*, la cual permite desarrollar un código para cargar un URDF y obtener su matriz dinámica. En el código implementado se definen los valores articulares de la posición por defecto y se indica la ruta hasta el URDF del robot Ares. Con esta información extrae del URDF la masa y la inercia rotacional del exoesqueleto. Mediante la librería *pinocchio* se crea un modelo del robot con la función *ccrba()* y se obtiene su matriz dinámica.

5.4.3.2 Ajustes de parámetros en TOWR

A continuación se van a comentar las variables que se han ajustado para el correcto funcionamiento del exoesqueleto en el simulador.

- **z_nominal_b**: representa la altura entre el suelo y el *base_link* (eslabón principal cuyos ejes principales se pueden ver en la Figura 5.20.B). Al definirlo menor que la altura entre el suelo y el *base_link* (extraída del URDF), se fuerza al robot a estar flexionado. Este ajuste evita que las piernas estén completamente extendidas y favorece la reconstrucción a partir de la posición del efector final que hace Xpp, ya que de lo contrario se realizarían movimientos fuera del espacio articular del robot. En esos casos la pierna adquiere una posición en la que no coincide del pie con el efector final (Figura 5.23).



(a) Exoesqueleto mal posicionado **(b)** Exoesqueleto bien posicionado

Figura 5.23: Comparación del exoesqueleto Ares antes y después de modificar el parámetro z_nominal_b.

En la Figura 5.23.A se puede ver como el pie de color azul claro no está sobre la esfera azul que representa la posición del efector final, como se ha explicado anteriormente esto se debe a que al estar el exoesqueleto completamente estirado no puede alcanzar la posición del efector final calculada por el simulador. Por el contrario, en la Figura 5.23.B los pies y los efectores finales (esferas azules) coinciden perfectamente.

- **y_nominal_b**: representa la distancia que se desea que esté desplazado el efector final sobre el eje Y. En este caso deseamos situarlo en el centro de la planta del pie, por lo tanto lo desplazamos una distancia **Lsep** hasta centrarlo.

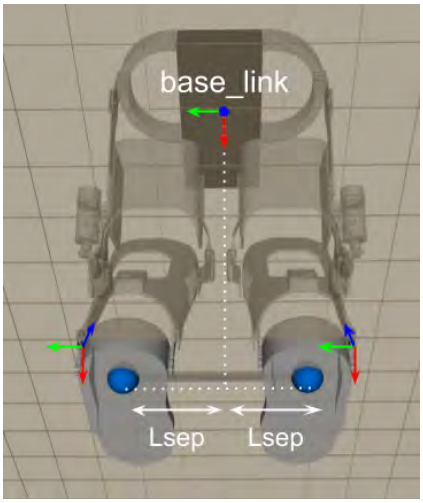


Figura 5.24: Representación de la distancia **Lsep** entre el *base_link* y el efector final sobre el eje Y.

- **max_dev_from_nominal_**: define las dimensiones de un paralelepípedo que se corresponde con el espacio de trabajo por el que se puede mover el efector final. Dado que los efectores finales (esferas azules) están conectados con los pies del URDF, establecen los límites articulares de los miembros inferiores.

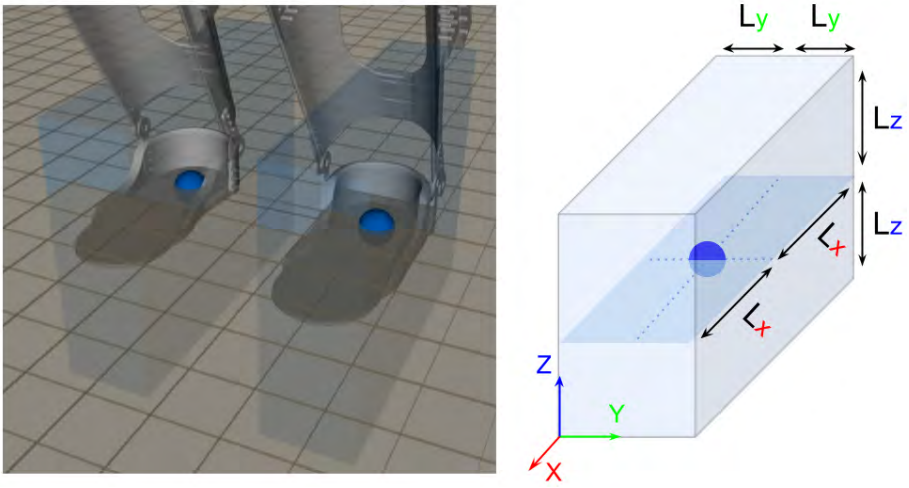


Figura 5.25: Captura y representación de los efectores finales (esferas azules) y el espacio de trabajo (paralelepípedo azul). Las variables Lx, Ly y Lz definen las dimensiones del paralelepípedo.

6 Resultados

En el siguiente apartado se van a mostrar los resultados obtenidos al realizar una serie de experimentos con los sistemas implementados en el desarrollo (Apartado 5) y se va a analizar dichos resultados.

6.1 Resultados en entorno de pruebas

Inicialmente se van a mostrar algunos de los resultados obtenidos durante la fase de pruebas con la escalera de cartón, para luego poder compararlos con los resultados con escaleras reales y verificar que los resultados concuerdan.

Figura 6.1: Entorno de pruebas con cámara de profundidad apuntando a los escalones.

Prueba	Frames	Altura Cámara (m)	Altura (m)	Profundidad (m)
1	3	1.0	0.1966	0.2855
2	5	1.0	0.1980	0.2886
3	5	0.9	0.1980	0.2913
4	5	0.8	0.1840	0.2940

Cuadro 6.1: Resultados obtenidos al realizar pruebas con la escalera de cartón descrita en el apartado 5.1.2. Las dimensiones reales de la escalera son **0.2** metros de altura y **0.3** metros de profundidad

Como se puede observar en en la tabla anterior, los resultados son bastante buenos. La media de los resultados son **0.194** metros de altura y **0.29** metros de profundidad, lo que se

aproxima bastante a los valores reales de la escalera (0.2 metros de altura y 0.3 metros de profundidad).

6.2 Resultados en entornos reales

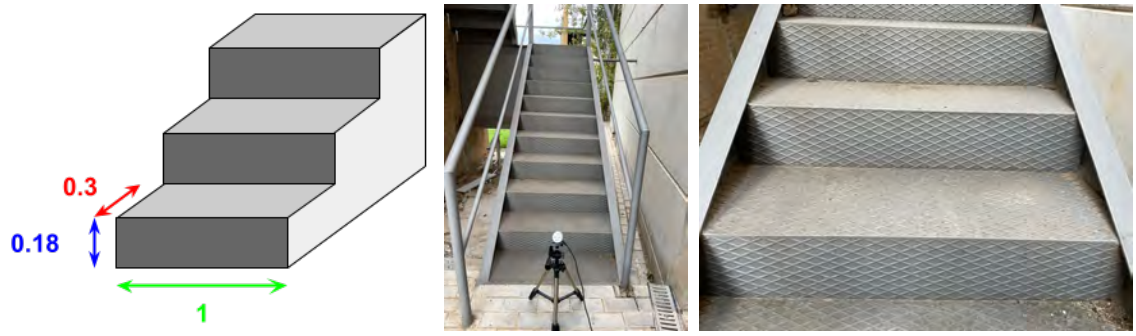
Seguidamente se van a mostrar los resultados obtenidos en entornos reales, tanto con escaleras de interior como de exterior. En la siguiente figura se pueden ver las escaleras elegidas para realizar las pruebas.

(a) Localización escaleras **(b)** Escalera 1 **(c)** Escalera 2

Figura 6.2: Localización de las escaleras e imágenes de dichas escaleras. La escalera 1 se trata de una escalera metálica exterior de emergencias y la escalera 2 consiste en una escalera de interior de mármol. Ambas escaleras se encuentran la EPS II de la Universidad de Alicante.

6.2.1 Escalera de exterior

La primera escalera que se capturó fue la de exterior (Figura 6.2.B), se trata de una escalera de emergencias anexa al edificio de la EPS II construida de metal.



(a) Dimensiones escalera 1 (m) **(b)** Captura **(c)** Vista de la cámara

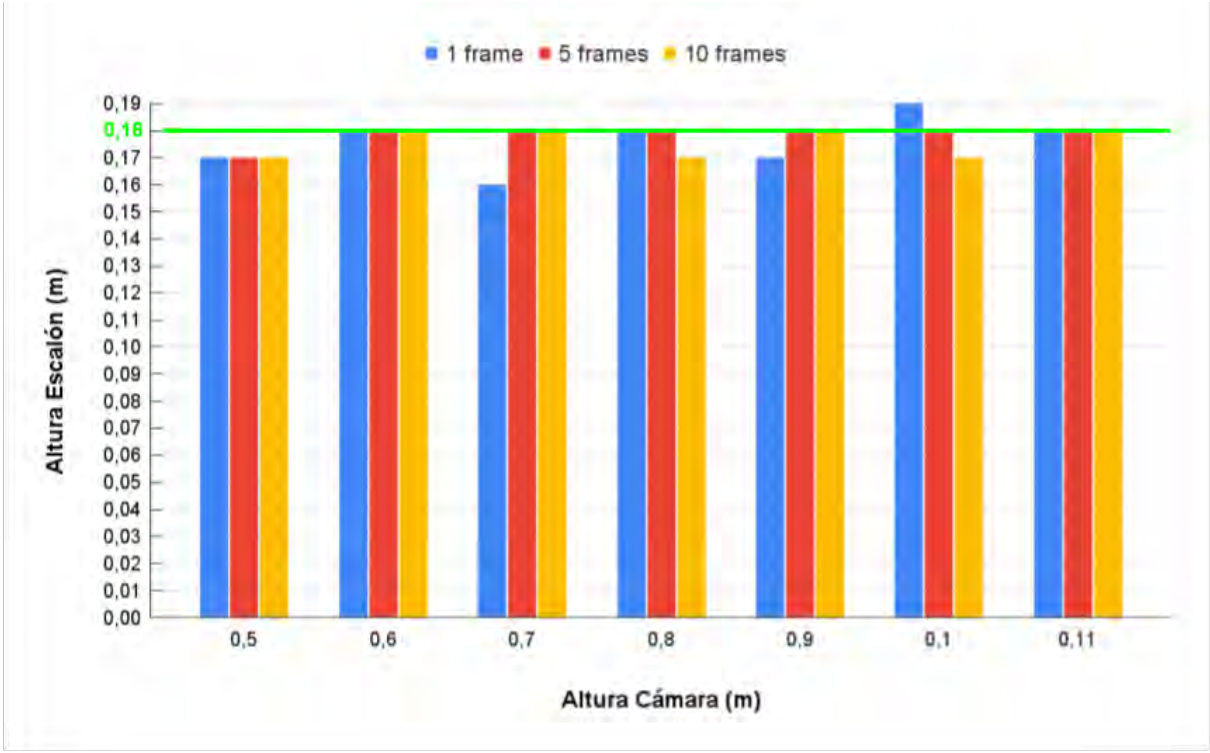
Figura 6.3: Dimensiones, posicionamiento y vista de la cámara de la escalera 1.

A continuación se muestra una tabla con los resultados de altura y profundidad tras capturar y procesar la escalera 1. Con el objetivo de determinar la altura a la que debe de estar la cámara y el número de *frames* que se deben capturar, se han realizado diferentes pruebas variando dichos parámetros.

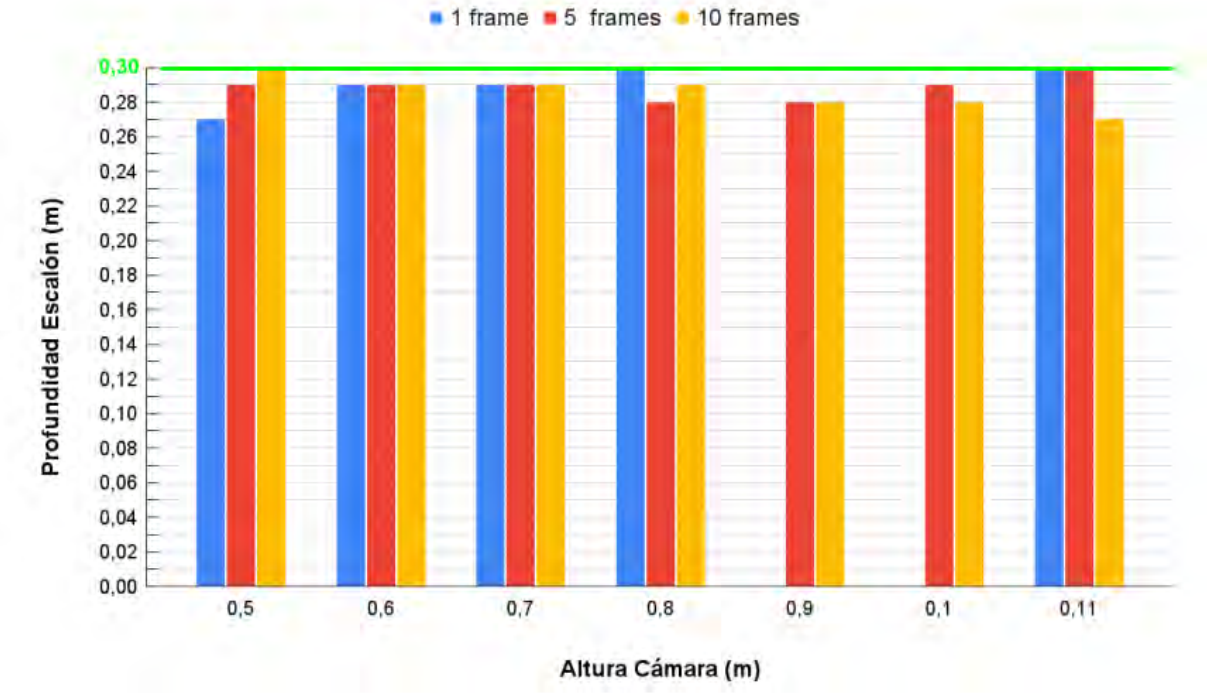
Altura Cámara	Frames	Altura	Profundidad
0.5	1	0.17	0.27
	5	0.17	0.29
	10	0.17	0.3
0.6	1	0.18	0.29
	5	0.18	0.29
	10	0.18	0.29
0.7	1	0.16	0.29
	5	0.18	0.29
	10	0.18	0.29
0.8	1	0.18	0.3
	5	0.18	0.28
	10	0.17	0.29
0.9	1	0.17	-
	5	0.18	0.28
	10	0.18	0.28
1	1	0.19	-
	5	0.18	0.29
	10	0.17	0.28
1.1	1	0.18	0.3
	5	0.18	0.3
	10	0.18	0.27

Cuadro 6.2: Tabla con los resultados obtenidos de la detección de la escalera 1 (Figura 6.2.B). Los valores de altura de la cámara, altura y profundidad están en metros. En las posiciones en las que aparece un guión ("-") no se pudo determinar un valor.

En el Cuadro 6.2, la primera columna corresponde a la altura a la que se ha colocado la cámara para realizar las capturas, la segunda a los frames que se han capturado y la tercera y cuarta columna contienen los valores de altura y profundidad del escalón que se han obtenido como resultado tras el procesado de la nube. En las posiciones en las que aparece un guión ("-") no se ha podido determinar la profundidad. Esto es debido a que como únicamente se ha tomado un *frame* de la escena y el valor ha sido descartado por erróneo, al no haber más *frames* de donde obtener información, resulta imposible determinar el valor. Como se puede apreciar en la tabla anterior, se ha variado la altura y el número de *frames* con el objetivo de determinar los valores idóneos. Para visualizar mejor los resultados se ha elaborado las siguientes gráficas:



(a) Altura de la cámara y del escalón

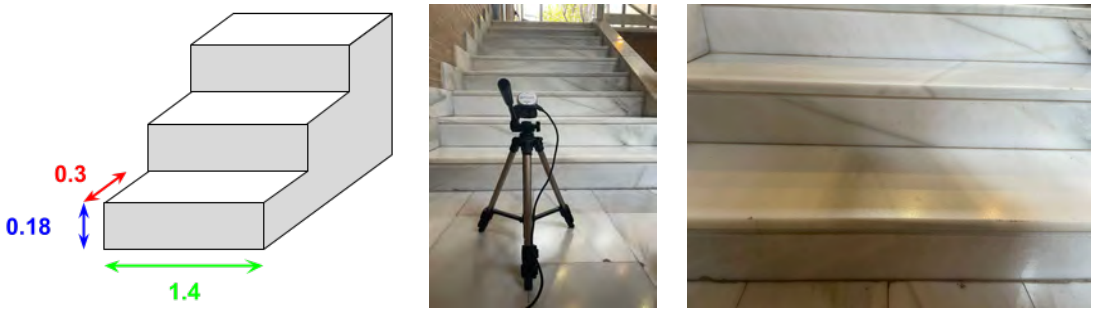


(b) Altura de la cámara y profundidad del escalón

Figura 6.4: Gráfica comparativa entre la altura de la cámara y las dimensiones de los escalones de la escalera 1 (altura y profundidad) obtenidas. La altura y profundidad real del escalón están marcadas en verde (0.18 y 0.3 respectivamente).

6.2.2 Escalera de interior

La segunda escalera que se capturó fue la de interior (Figura 6.2.C), se trata de una escalera de uso general en el interior del edificio de la EPS II construida de mármol.



(a) Dimensiones escalera 2 (m) **(b)** Captura **(c)** Vista de la cámara

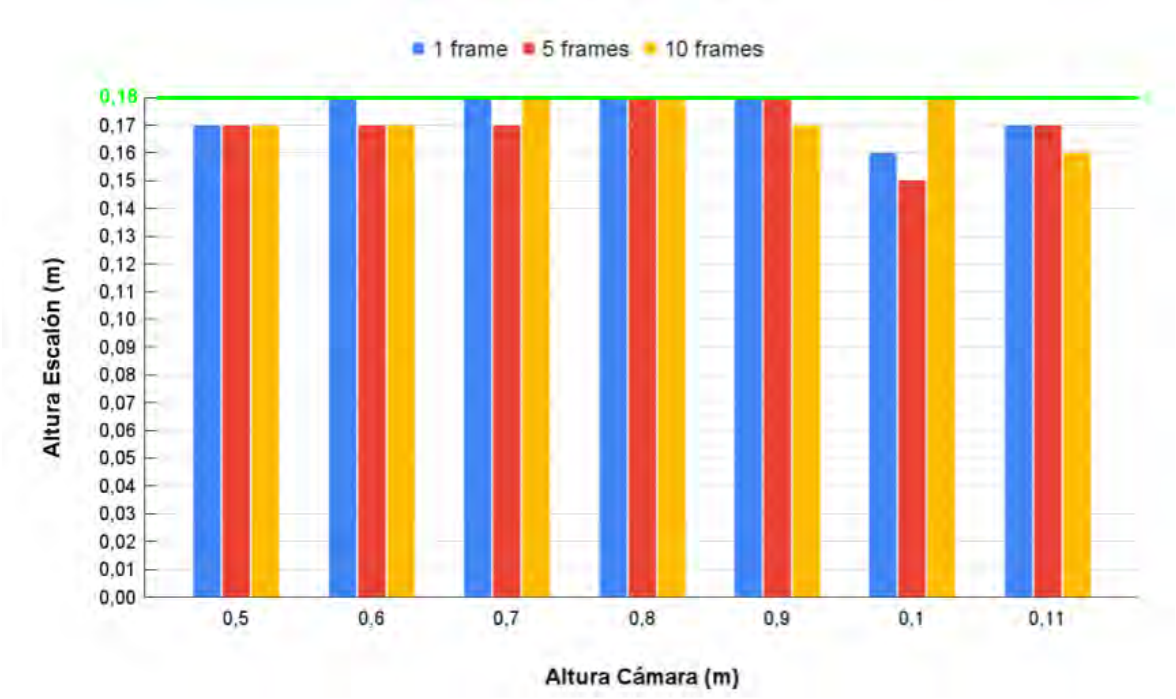
Figura 6.5: Dimensiones, posicionamiento y vista de la cámara de la escalera 2.

A continuación se muestra una tabla con los resultados de altura y profundidad tras capturar y procesar la escalera 2.

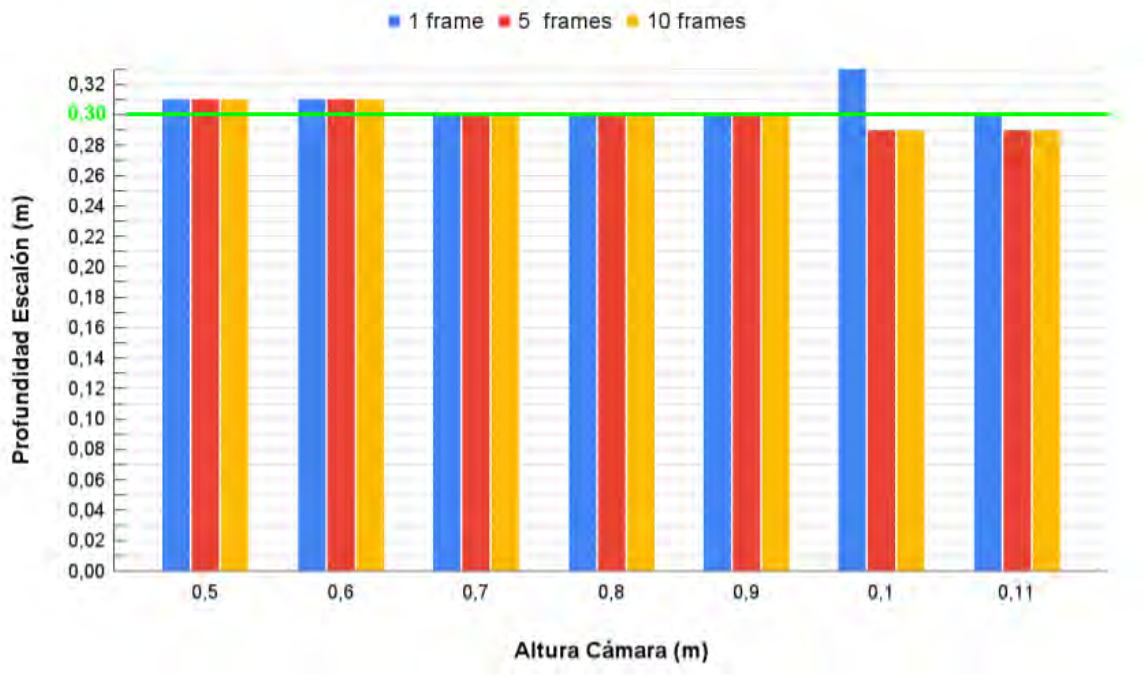
Altura Cámara	Frames	Altura	Profundidad
0.5	1	0.17	0.31
	5	0.17	0.31
	10	0.17	0.31
0.6	1	0.18	0.31
	5	0.17	0.31
	10	0.17	0.31
0.7	1	0.18	0.3
	5	0.17	0.3
	10	0.18	0.3
0.8	1	0.18	0.3
	5	0.18	0.3
	10	0.18	0.3
0.9	1	0.18	0.3
	5	0.18	0.3
	10	0.17	0.3
1	1	0.16	0.33
	5	0.15	0.29
	10	0.18	0.29
1.1	1	0.17	0.3
	5	0.17	0.29
	10	0.16	0.29

Cuadro 6.3: Tabla con los resultados obtenidos de la detección de la escalera 2 (Figura 6.2.C). Los valores de altura de la cámara, altura y profundidad están en metros.

En el Cuadro 6.3, el significado de cada columna es similar al explicado para la tabla anterior (Cuadro 6.2). Para visualizar mejor los resultados se ha elaborado las siguientes gráficas:



(a) Altura de la cámara y del escalón



(b) Altura de la cámara y profundidad del escalón

Figura 6.6: Gráfica comparativa entre la altura de la cámara y las dimensiones de los escalones de la escalera 2 (altura y profundidad) obtenidas. La altura y profundidad real del escalón están marcadas en verde (0.18 y 0.3 respectivamente).

6.3 Resultados redondeo decimales

Como se ha comentado en el apartado 5.2.4, se ha realizado un redondeo de las coordenadas de las posiciones de los puntos a 2 decimales, por lo que se reduce la precisión de la detección a dos decimales. A continuación se muestran los resultados de algunas pruebas que se realizaron al aumentar el número de decimales en el redondeo realizado.

Nº Decimales	Altura Cámara (m)	Frames	Altura Detectada (m)	Profundidad Detectada (m)	Tiempo Procesamiento (seg)
2	0.6	5	0.18	0.29	12.61
3	0.6	5	0.176	0.275	109.68
4	0.5	5	0.2067	0.3169	260.00

Cuadro 6.4: Resultados obtenidos tras aumentar el número de decimales del redondeo realizado en el procesamiento de la nube.

6.4 Resultados TOWR

La trayectoria generada tras cargar el URDF y ajustar parámetros se pueden ver en la siguiente secuencia de imágenes en las que se puede ver la progresión paso a paso de la trayectoria.

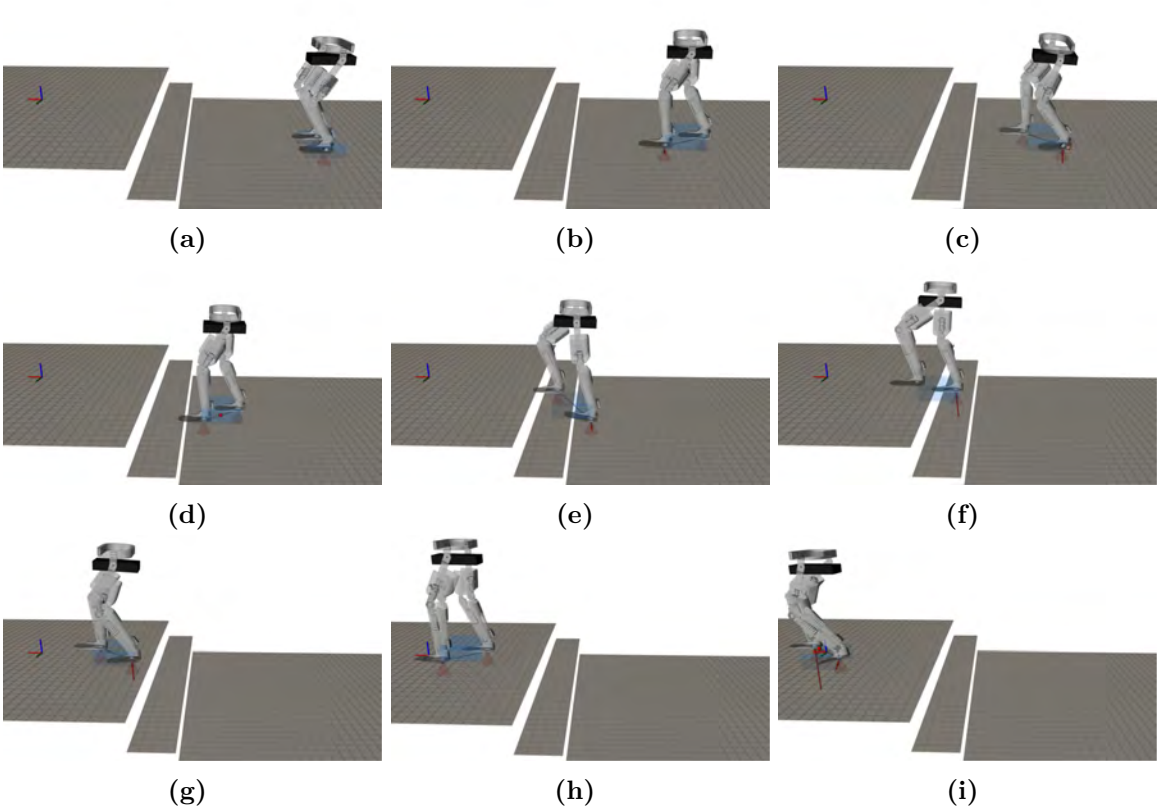
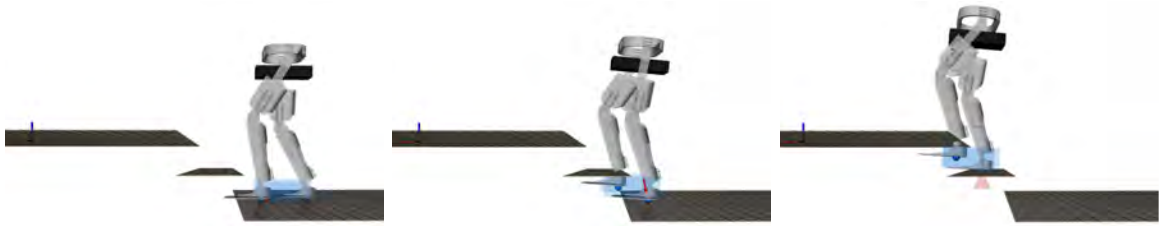


Figura 6.7: Secuencia de imágenes de la trayectoria generada con TOWR.

Como se puede apreciar, se genera una trayectoria que alcanza la posición deseada. Aunque la trayectoria no es del todo correcta ya que TOWR únicamente tiene en cuenta la posición de los efectores finales y no toda la planta del pie del exoesqueleto. Es por ello que al subir las escaleras, durante unos períodos muy pequeños de tiempo, se sitúan los pies en configuraciones erróneas atravesando el terreno.



(a) (b) (c)

Figura 6.8: Captura de los errores al generar las trayectorias. En la Figura A coloca la punta del pie por el interior de la contrahuella del escalón, en las Figuras B y C atraviesa con el tobillo el escalón.

Por tanto aunque las posiciones de los efectores finales generadas por TOWR son correctas, la trayectoria generada no lo es del todo, por lo que la visualización del exoesqueleto falla en algunos puntos por un breve instante. A continuación se pueden apreciar las gráficas generadas de la la trayectoria anteriormente mostrada (Figura 6.7).

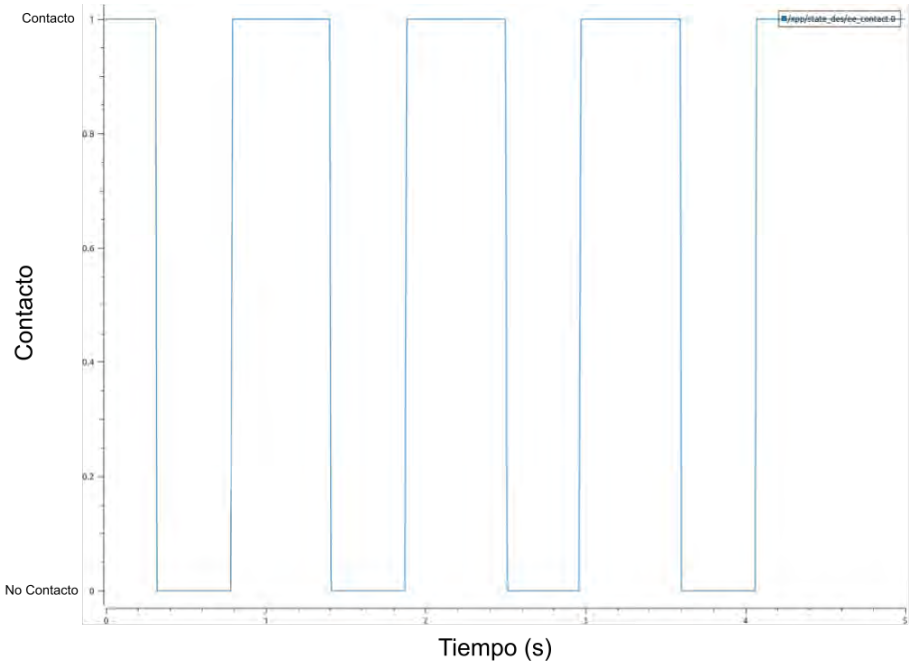


Figura 6.9: Gráfica del contacto del pie izquierdo

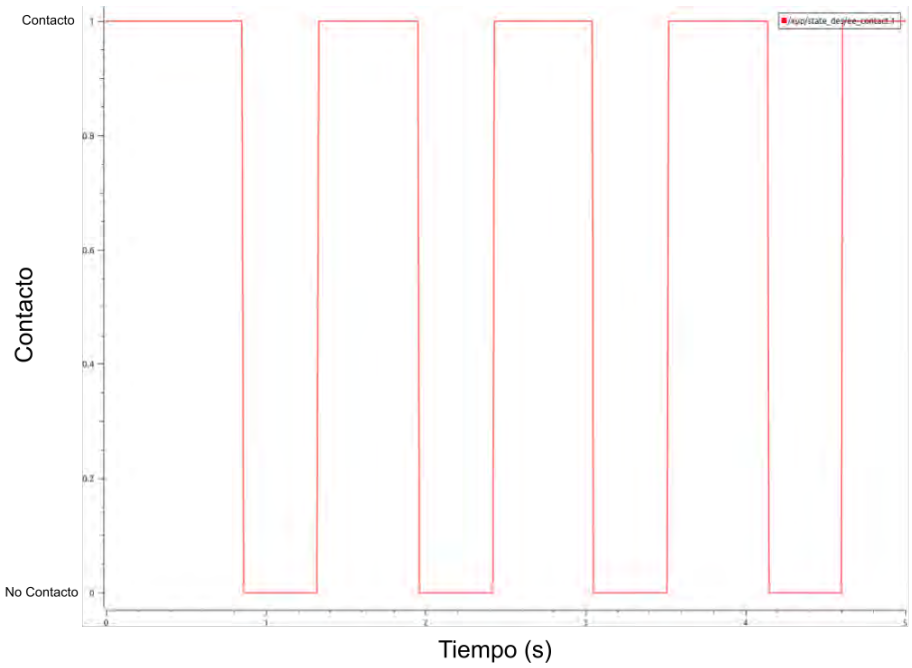


Figura 6.10: Gráfica del contacto del pie derecho

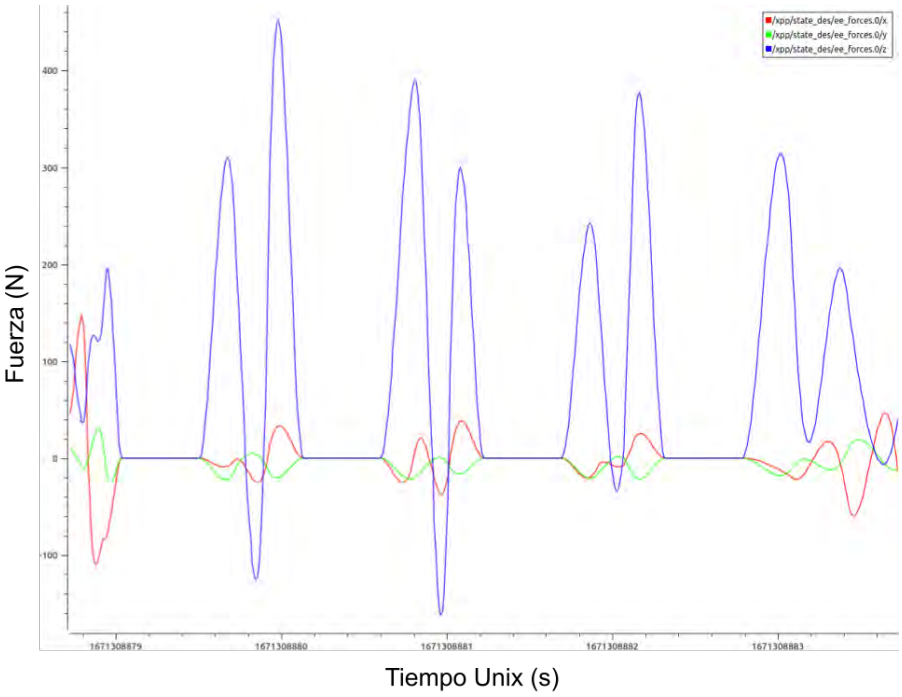


Figura 6.11: Gráfica de las fuerzas del pie izquierdo.

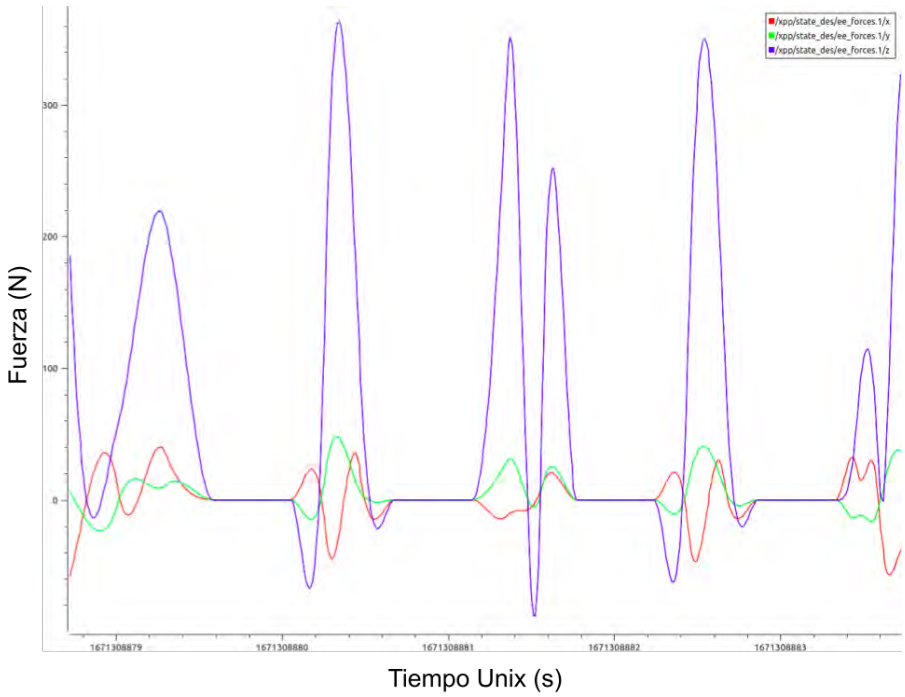


Figura 6.12: Gráfica de las fuerzas del pie derecho.

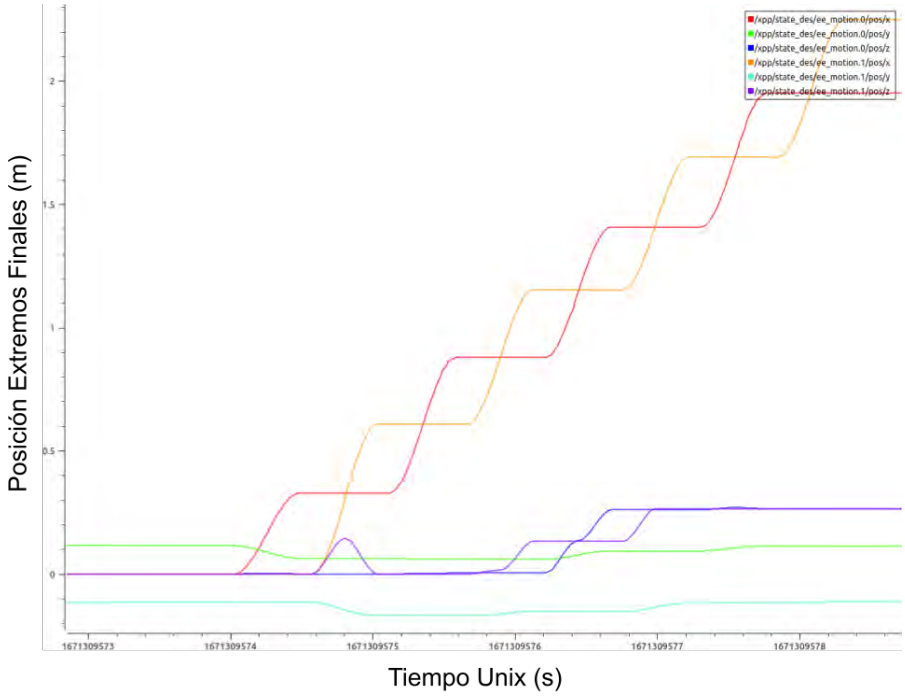


Figura 6.13: Gráfica de la posición de los extremos finales de ambos pies.

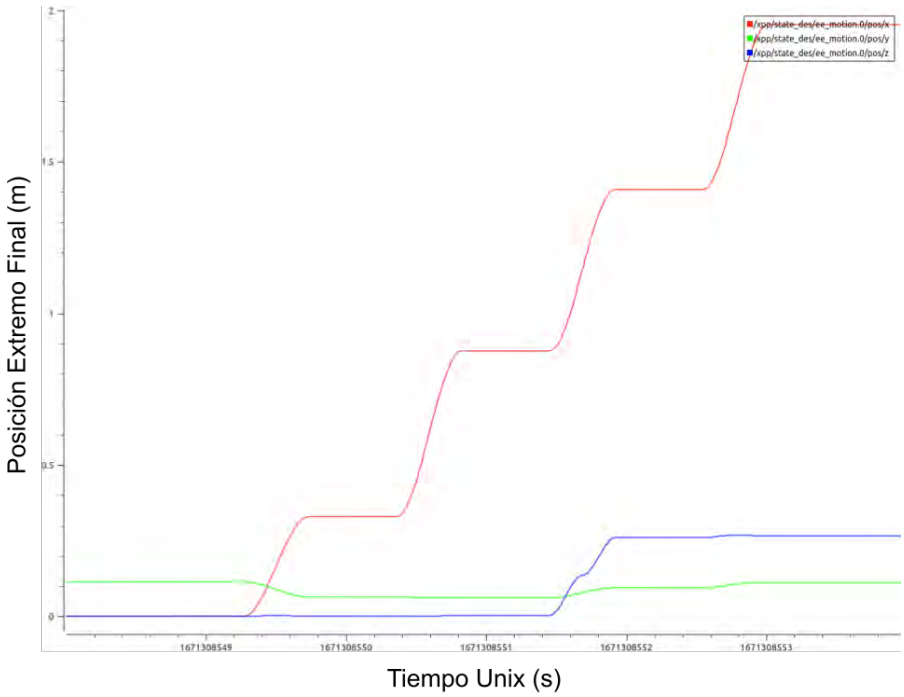


Figura 6.14: Gráfica de la posición del extremo final del pie izquierdo individualmente.

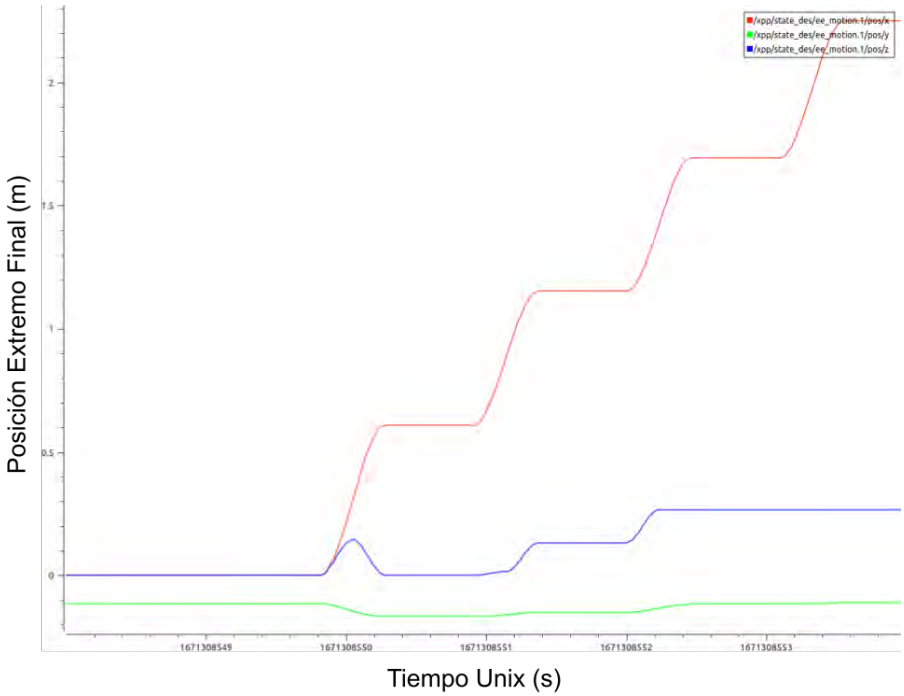


Figura 6.15: Gráfica de la posición del extremo final del pie derecho individualmente.

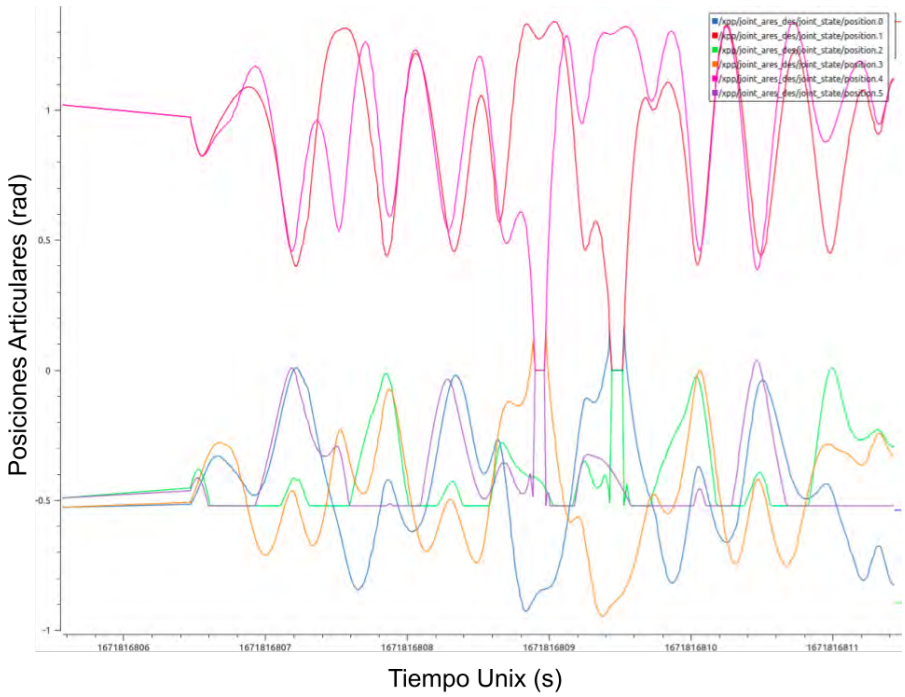
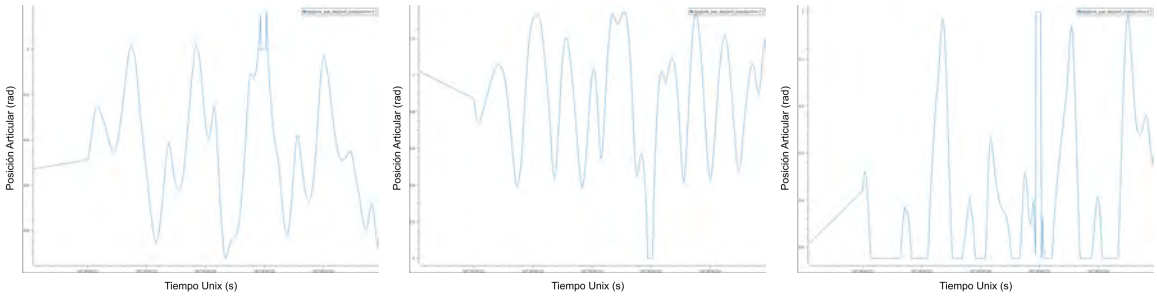
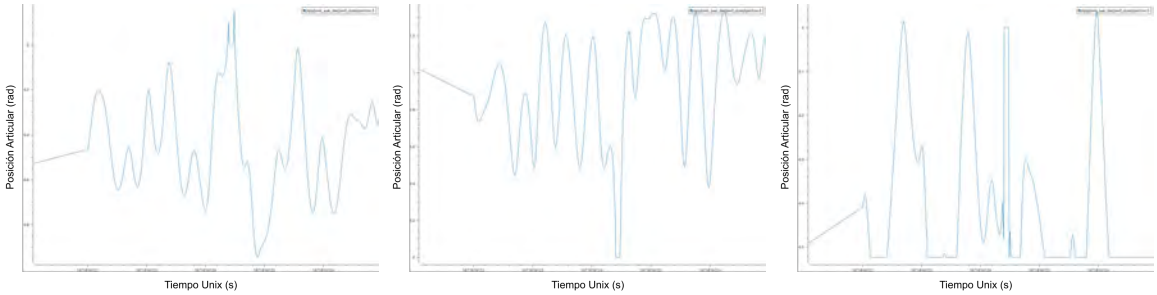


Figura 6.16: Gráfica de los valores articulares conjuntamente



(a) Position.0: Muslo derecho **(b)** Position.1: Pierna derecha **(c)** Position.2: Pie derecho



(d) Position.3: Muslo izquierdo **(e)** Position.4: Pierna izquierda **(f)** Position.5: Pie izquierdo

Figura 6.17: Gráfica de los valores articulares individualmente

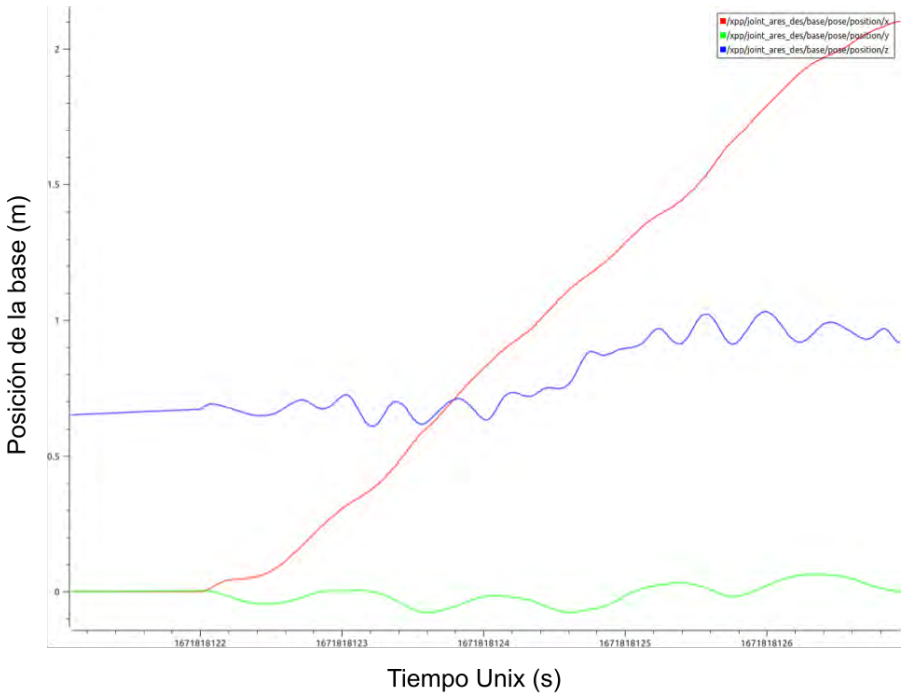


Figura 6.18: Gráfica de la posición de la base durante el movimiento.

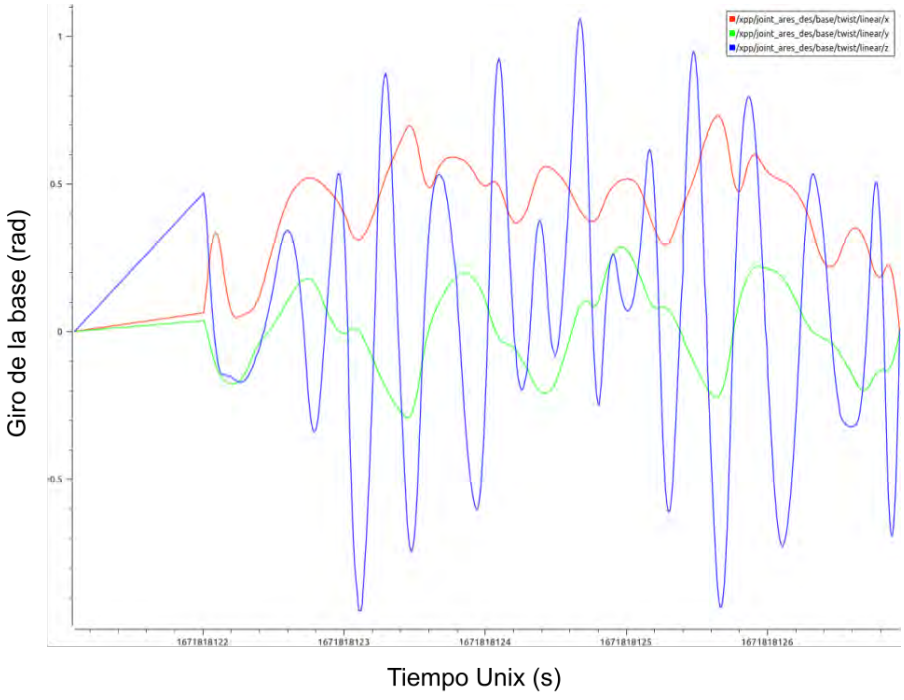


Figura 6.19: Gráfica del giro de la base durante el movimiento.

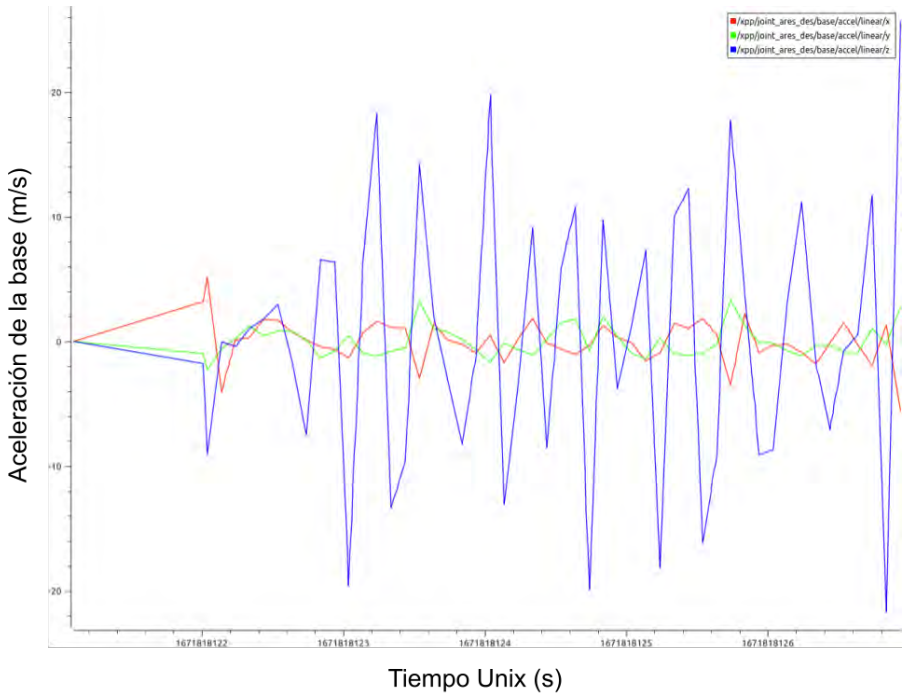


Figura 6.20: Gráfica de la aceleración de la base durante el movimiento.

En el siguiente enlace se puede acceder a un vídeo del exoesqueleto Ares realizando la trayectoria generada por el simulador TOWR para subir una escalera `https://youtu.be/Voi5RI1ZNqk`.

7 Análisis de resultados

En este apartado se van a evaluar los resultados obtenidos durante las pruebas realizadas. Para mayor claridad, se van a evaluar por separado la captura y procesamiento de las nubes de puntos y la generación de trayectorias en TOWR.

7.1 Análisis captura y procesamiento

En lo referente al procesamiento de la nube de puntos de escaleras reales, para unos escalones con altura 0.18 metros y profundidad 0.3 metros, se ha logrado obtener unos resultados de **0.175 ± 0.03 metros** de altura y **0.29 ± 0.03 metros** de profundidad.

En los resultados del procesamiento de la nube, al aumentar el número de decimales del redondeo (Apartado 6.3), se ha podido comprobar que los resultados empeoran ya que, aunque los puntos están definidos con mayor precisión, hay menor número de puntos que coinciden en las mismas posiciones, lo que produce un aumento de los puntos con los que se ha de trabajar. Además, al haber mayor número de puntos, la elección de los mejores se hace más compleja. Como consecuencia aumenta considerablemente el tiempo de procesamiento y se reduce la precisión de los resultados. Es por ello que en el caso de este trabajo se ha considerado que el mejor número de decimales es 2, debido a una mayor precisión de los resultados y a un tiempo de procesamiento menor.

En el procesamiento de la nube, más concretamente en el redondeo de las coordenadas (Apartado 5.2.4), se ha comentado que se ha realizado un redondeo de las coordenadas de las posiciones de los puntos, por lo que se reduce la precisión de la detección a dos decimales. Tras realizar pruebas aumentando el redondeo a un mayor número de decimales, se ha comprobado que el tiempo de procesamiento aumenta considerablemente y que la precisión 17.38

En cuanto al número de *frames* que se ha de tomar de cada escena, se ha podido comprobar que aunque la opción más rápida es tomar un único *frame*, también puede resultar insuficiente en caso de errores en la captura o el procesamiento. Por otro lado, capturar un gran número de *frames* (10 en el caso de este trabajo) aumenta el tiempo y aunque en ocasiones logra resultados mejores que con 5 *frames*, en la mayoría de los casos se obtienen los mismos resultados o difieren en 0.01 metros. Es por ello que se ha podido determinar que es necesario capturar más de un *frame* para hacer más robusto el sistema, pero el número exacto depende de la precisión requerida y el tiempo disponible.

Otro aspecto con el que se ha experimentado es con la altura de la cámara al realizar las capturas, esta se ha variado desde los 0.5 metros hasta los 1.1 metros. A partir de los experimentos realizados y los resultados obtenidos, se ha podido determinar que, estando

siempre dentro del rango de captura de las especificaciones de la cámara, la altura a la que se realizan las capturas no es tan relevante, al contrario que la orientación de la cámara y el encuadre de la escena para que aparezcan fácilmente detectables los elementos requeridos para el procesamiento de los escalones como los bordes y las contrahuellas.

7.2 Análisis generación de trayectorias

En lo relativo a la generación de trayectorias con TOWR, en las gráficas mostradas en el apartado 6.4, se puede apreciar los resultados de la mejor trayectoria que se ha podido generar. El exoesqueleto alcanza la posición objetivo subiendo las escaleras. Como se ha comentado anteriormente en dicho apartado, se producen algunos errores debido a que TOWR únicamente tiene en cuenta la posición de los efectores finales y no toda la superficie de la plata del pie del exoesqueleto. Es por ello que al visualizar la trayectoria hay instantes en los que el exoesqueleto atraviesa los escalones (Figura 6.8).

8 Conclusiones y trabajos futuros

Como conclusión, se han alcanzado los objetivos definidos en el apartado de objetivos (apartado 2). Se ha logrado controlar la cámara de profundidad, realizar capturas de escenas con escaleras, obtener las dimensiones con cierta precisión, recrear la escalera en el simulador TOWR y generar las trayectorias para subirla. Es por ello que la realización de este trabajo ha permitido investigar sobre las posibilidades que ofrecen las cámaras de profundidad en tareas de detección de obstáculos, la extracción de información de nubes de puntos y la planificación de movimientos para exoesqueletos con el propósito de ampliar sus capacidades. También se ha comprendido la importancia de dotar de robustez a los sistemas y algoritmos de detección para poder reducir el número de errores y hacerlos más fiables.

Como trabajo futuro se plantea realizar el cálculo de la distancia que separa la cámara y el primer escalón detectado para situarlo con precisión en TOWR. De este modo, la generación de la trayectoria sería más acorde con la realidad. También se sugiere la posibilidad de añadir sensores al exoesqueleto para aumentar la precisión de las detecciones y tener un mayor conocimiento del entorno y su posición en él. Además de escaleras, se plantea incluir el procesamiento y extracción de características de otras barreras arquitectónicas como rampas, que junto con las escaleras son las dos estructuras principales para comunicar diferentes alturas. Asimismo se propone seguir mejorando el algoritmo de procesamiento de la nube de puntos para hacerlo más robusto, pudiendo reconocer si en la escena hay escaleras o no y seguir trabajando en la generación de trayectorias en TOWR, para resolver los problemas de atravesar el terreno con el URDF del exoesqueleto. Por último se plantea la posibilidad de una vez resueltos los problemas de las trayectorias con TOWR, enviar la información a unos motores reales y ejecutar los movimientos para posteriormente colocarlos en el exoesqueleto ARES y proseguir con el desarrollo de ese proyecto.

Bibliografía

Aposhian, A. (2017). *perception$_p$cl.*. GitHub.

Asbeck, A. T., Schmidt, K., y Walsh, C. J. (2015). Soft exosuit for hip assistance. *Robotics and Autonomous Systems*, *73*, 102-110. Descargado de `https://www.sciencedirect.com/science/article/pii/S0921889014002103` (Wearable Robotics) doi: https://doi.org/10.1016/j.robot.2014.09.025

Bags. (2022). Descargado de `http://wiki.ros.org/Bags`

Bao, G., Pan, L., Fang, H., Wu, X., Yu, H., Cai, S., … Wan, Y. (2019). Academic review and perspectives on robotic exoskeletons. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *27*(11), 2294-2304. doi: 10.1109/TNSRE.2019.2944655

Bernat, L. (2022). *towr.* `https://github.com/lbernat`. GitHub.

Bernat Iborra, L. (2022). *Planificación de trayectorias de un exoesqueleto de miembro inferior para rehabilitación.* Descargado de `http://rua.ua.es/dspace/handle/10045/124647`

Betts, J. T. (1998). Survey of numerical methods for trajectory optimization. *Journal of guidance, control, and dynamics*, *21*(2), 193–207.

BOE. (2006). Real decreto 314/2006, de 17 de marzo por el que se aprueba el código técnico de la edificación. *Boletín Oficial del Estado*, 926–929.

Bogue, R. (2009). Exoskeletons and robotic prosthetics: a review of recent developments. *Industrial Robot*, *36*(5), 421-427. doi: https://doi.org/10.1108/01439910910980141

Béchet, E., Cuilliere, J.-C., y Trochu, F. (2002). Generation of a finite element mesh from stereolithography (stl) files. *Computer-Aided Design*, *34*(1), 1-17. Descargado de `https://www.sciencedirect.com/science/article/pii/S0010448500001469` doi: https://doi.org/10.1016/S0010-4485(00)00146-9

Carpentier, J., Valenza, F., Mansard, N., y cols. (2015–2021). *Pinocchio: fast forward and inverse dynamics for poly-articulated systems.* https://stack-of-tasks.github.io/pinocchio.

Chen, B., Ma, H., Qin, L.-Y., Gao, F., Chan, K.-M., Law, S.-W., … Liao, W.-H. (2016). Recent developments and challenges of lower extremity exoskeletons. *Journal of Orthopaedic Translation*, *5*, 26-37. Descargado de `https://www.sciencedirect.com/science/article/pii/S2214031X15000716` (Special Issue: Orthopaedic Biomaterials and Devices) doi: https://doi.org/10.1016/j.jot.2015.09.007

Doronhi. (2021). *Ros wrapper for intel® realsense™ devices (build 2.3.2).* `https://github.com/IntelRealSense/realsense-ros/releases/tag/2.3.2`. GitHub.

Exo-h3-technaid-exoesqueleto. (s.f.). Descargado de `https://www.technaid.com/wp-content/uploads/2019/05/Exo-H3-Technaid-Exoesqueleto.png`

Faconti, D. (2016). *Plotjuggler.* `https://github.com/facontidavide/PlotJuggler`. GitHub.

Featherstone, R. (2014). *Rigid body dynamics algorithms.* Springer.

Garcia, L. (2008). Els isòpodes terrestres (crustacea: Isopoda: Oniscidea) del parc natural de l'illa de sa dragonera (illes balears, mediterrània occidental). *Bolletí de la Societat d'Història Natural de les Balears*, 203–223.

Gutmann, J.-S., Fukuchi, M., y Fujita, M. (2008). 3d perception and environment map generation for humanoid robot navigation. *The International Journal of Robotics Research*, *27*(10), 1117-1134. doi: 10.1177/0278364908096316

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., … Oliphant, T. E. (2020, septiembre). Array programming with NumPy. *Nature*, *585*(7825), 357–362. Descargado de `https://doi.org/10.1038/s41586-020-2649-2` doi: 10.1038/s41586-020-2649-2

Hart, S., Dinh, P., y Hambuchen, K. (2015). The affordance template ros package for robot task programming. En *2015 ieee international conference on robotics and automation (icra)* (p. 6227-6234). doi: 10.1109/ICRA.2015.7140073

Huo, W., Mohammed, S., Moreno, J. C., y Amirat, Y. (2016). Lower limb wearable robots for assistance and rehabilitation: A state of the art. *IEEE Systems Journal*, *10*(3), 1068-1081. doi: 10.1109/JSYST.2014.2351491

Intel realsense lidar camera l515. (s.f.). Descargado de `https://www.intelrealsense.com/lidar-camera-l515/`

Kang, Y., Kim, D., y Kim, K. (2019). Urdf generator for manipulator robot. En *2019 third ieee international conference on robotic computing (irc).* doi: 10.1109/IRC.2019.00101

Kida, Y., Kagami, S., Nakata, T., Kouchi, M., y Mizoguchi, H. (2004). Human finding and body property estimation by using floor segmentation and 3d labelling. En *2004 ieee international conference on systems, man and cybernetics (ieee cat. no.04ch37583)* (Vol. 3, p. 2924-2929 vol.3). doi: 10.1109/ICSMC.2004.1400777

LLC, M. (1999). *The pcd (point cloud data) file format.* Descargado 2022-11-11, de `http://pointclouds.org/documentation/tutorials/pcd_file_format`

Lo, H. S., y Xie, S. Q. (2012). Exoskeleton robots for upper-limb rehabilitation: State of the art and future prospects. *Medical Engineering Physics*, *34*(3), 261-268. Descargado de `https://www.sciencedirect.com/science/article/pii/S1350453311002694` doi: https://doi.org/10.1016/j.medengphy.2011.10.004

Luo, R. C., Hsiao, M., y Liu, C.-W. (2013). Multisensor integrated stair recognition and parameters measurement system for dynamic stair climbing robots. En *2013 ieee international conference on automation science and engineering (case)* (p. 318-323). doi: 10.1109/CoASE.2013.6654026

OpenCV, T. (2022). *Opencv.* Descargado de `https://opencv.org/about/`

Oßwald, S., Gutmann, J.-S., Hornung, A., y Bennewitz, M. (2011). From 3d point clouds to climbing stairs: A comparison of plane segmentation approaches for humanoids. En *2011*

11th ieee-ras international conference on humanoid robots (p. 93-98). doi: 10.1109/Humanoids .2011.6100836

Python. (2022). Descargado de `https://www.python.org/`

Qian, X., y Ye, C. (2014, 04). Ncc-ransac: A fast plane extraction method for 3-d range data segmentation. *IEEE transactions on cybernetics*, *44*. doi: 10.1109/TCYB.2014.2316282

Rosbag. (2020). Descargado de `http://wiki.ros.org/rosbag`

Ros-gazebo. una valiosa herramienta de vanguardia para el desarrollo de la robótica. (s.f.). , *10*. Descargado de `https://hemeroteca.unad.edu.co/index.php/publicaciones-e-investigacion/article/view/1593` doi: 10.22490/25394088.1593

Ros/ introduction. (2018). Descargado de `http://wiki.ros.org/ROS/Introduction`

Rusu, R. B., y Cousins, S. (2011, May 9-13). 3D is here: Point Cloud Library (PCL). En *IEEE International Conference on Robotics and Automation (ICRA).* Shanghai, China: IEEE.

Sato, D. (2022). $perception_pcl.$. GitHub.

Tantichattanont, P., Songschon, S., y Laksanacharoen, S. (2007). Quasi-static analysis of a leg-wheel hybrid vehicle for enhancing stair climbing ability. En *2007 ieee international conference on robotics and biomimetics (robio)* (p. 1601-1605). doi: 10.1109/ROBIO.2007 .4522404

Urdf transmissions. (2017). Descargado de `http://wiki.ros.org/urdf/XML/Transmission`

Using xacro to clean up a urdf file. (2021). Descargado de `http://wiki.ros.org/urdf/Tutorials/Using%20Xacro%20to%20Clean%20Up%20a%20URDF%20File#Leg_macro`

Van Rossum, G., y Drake Jr, F. L. (1995). *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam.

Winkler, A. (2021). *Towr.* `https://github.com/ethz-adrl/towr`. GitHub.

Winkler, A. W. (2017). *Xpp - A collection of ROS packages for the visualization of legged robots.* Descargado de `https://doi.org/10.5281/zenodo.1037901` doi: 10.5281/zenodo .1037901

Winkler, A. W., Bellicoso, D. C., Hutter, M., y Buchli, J. (2018, July). Gait and trajectory optimization for legged systems through phase-based end-effector parameterization. *IEEE Robotics and Automation Letters (RA-L)*, *3*, 1560-1567. doi: 10.1109/LRA.2018.2798285

Woo, S., Shin, J., Lee, Y. H., Hun Lee, Y., Lee, H., Kang, H., … Moon, H. (2019). Stair-mapping with point-cloud data and stair-modeling for quadruped robot. En *2019 16th international conference on ubiquitous robots (ur)* (p. 81-86). doi: 10.1109/URAI.2019.8768786

Yang, N. (1890). *Apparatus for facilitating walking, running, and jumping* (no 420179).

Lista de Acrónimos y Abreviaturas

CoM	Center of Mass.
EPS	Escuela Politécnica Superior.
FPS	Frames per Second.
Hardiman	Human Augmentation Research and Development Investigation.
HURO	Human Robotics.
IMU	Inertial Measurement Unit.
LiDAR	Light Detection and Ranging.
NLP	NonLinear Problems.
PCD	Point Cloud Data.
PCL	Point Cloud Library.
RANSAC	RANdom SAmple Consensus.
RGB	Red, Green and Blue color representation.
ROS	Robot Operating System.
RVIZ	Robot Visualizer.
STL	Standard Triangule Language.
TO	Trajectory Optimization.
TOWR	Trajectory Optimizer for Walking robots.
URDF	Unified Robotics Description Format.
XML	eXtensible Markup Language.