

Optimización de Trayectorias para la locomoción de robots con patas

Ramón Calvo – Víctor González – Sheila Sánchez

15 de enero de 2021

Vídeo parte práctica: https://youtu.be/KKZSU_rd8V0

Índice

1. Introducción	2
1.1. El panorama mas amplio	2
2. Modelos dinámicos para sistemas con patas	3
2.1. <i>Rigid Body Dynamics</i>	4
2.2. <i>Single Rigid Body Dynamics</i>	5
2.3. <i>Linear Inverted Pendulum Model</i>	7
3. Física de la locomoción con patas	9
4. Planificación de locomoción tradicional con patas	9
5. Optimización de Trayectorias	10
5.1. Métodos	10
6. Algoritmos de planificación de movimiento	12
7. Ejemplo de aplicación a robots cuadrúpedos	15
7.1. Formulación	16
7.2. Resultados	22
8. Ejemplo aplicado a robots bípedos	23
9. Optimización de Trayectorias en el futuro	25

1. Introducción

Los robots han comenzado a convertirse en algo común en la sociedad actual. A pesar de haber visto una gran mejora tecnológica para los robots voladores y rodantes, un modo de transporte que aun no ha dado el salto es la locomoción con patas. Esta discrepancia es desafortunada ya que la locomoción con patas tiene una variedad de ventajas sobre volar o rodar.

Una ventaja significativa es que no debe existir un camino continuo entre el inicio y la meta, sino que las posiciones discretas para colocar los pies son suficientes para moverse. Esta capacidad permite a los sistemas con patas, subir escaleras, atravesar escombros en áreas de terremotos o trabajar bajo tierra en terrenos muy desestructurados. Además, las maquinas con patas utilizan el entorno para moverse, lo que generalmente permite cargas útiles mas grandes con menos esfuerzo en comparación con las maquinas voladoras.

Estas capacidades de las máquinas con patas tienen el potencial de beneficiar a la sociedad. Algunos ejemplos son: robot en hogares con escalera; búsqueda y rescate en situaciones peligrosas; o utilizarlos en simbiosis con los humanos, para ayudar a los pacientes paralizados a recuperar la movilidad.

1.1. El panorama mas amplio

Una tarea tan compleja requiere varias capacidades que deben interactuar para generar, por ejemplo, un movimiento de caminar. Esto es responsabilidad de tres componentes: sentir, actuar y planificar. En este trabajo nos centraremos en la planificación.



Figura 1: Capacidades individuales para generar movimientos para sistemas autónomos.

- **Sentir:** la primera tarea de un agente autónomo es percibir su entorno. Para mantener el equilibrio mientras camina, el robot se basa en una IMU y la estimación del estado para determinar su posición global y como esta orientado su cuerpo. Necesita saber que obstáculos del terreno se encuentran frente a el y se deben evitar. Las señales procedentes

de cámaras o escáneres láser deben analizarse para construir un mapa preciso del entorno.

- **Actuar:** una vez percibido el entorno y el propio estado, se planifica un movimiento que posteriormente se ejecuta en el robot. Esta ejecución incluye aplicar un torque a las articulaciones para generar un movimiento del sistema. El software calcula la corriente correcta del motor (o la apertura de la válvula hidráulica) para crear el par deseado. También se colocan los controladores que deben observar continuamente el estado actual del robot y adaptar los pares de articulación en consecuencia. Este problema de generar pares de torsión, ha sido ampliamente estudiado y existen muchas soluciones útiles. La mayoría de los enfoques se basan en el uso de la dinámica del cuerpo rígido (RBD) mientras se imponen restricciones adicionales [1] o en el seguimiento directo de las fuerzas del efecto final planificadas [2].
- **Planificar:** el vínculo entre la detección del entorno y la ejecución de un movimiento se denomina planificación de movimiento. Si todo el bucle de la figura 1 se repite lo suficientemente rápido y el plan se produce mediante la optimización de la dinámica del robot, el método se denomina *Model Predictive Control* (MPC). Esta optimización frecuente es bastante peculiar, ya que sin ningún método adicional el sistema obtiene una variedad de capacidades reactivas que son cruciales para implementar robots en entornos del mundo real. Sin embargo, la planificación de movimiento realizada en cada ciclo de la Figura 1 es un bloque de construcción critico. Si el plan de movimiento ya es físicamente inviable, es imposible que incluso los mejores controladores o una estructura MPC lo ejecuten.

2. Modelos dinámicos para sistemas con patas

Se puede utilizar una variedad de modelos matemáticos para diseñar algoritmos eficientes para la planificación de la locomoción con patas. Un plan de movimiento se considera físicamente correcto según un modelo dado, si las ecuaciones de modelado se cumplen en todo momento. Estos modelos cuantifican la relación entre una entrada u con el estado actual x del sistema y el cambio de estado resultante x' . Esto se puede modelar mediante la ecuación diferencial:

$$x'(t) = \mathcal{F}(x(t), u(t))$$

A continuación, se presentan diferentes modelos dinámicos para representar robots con patas que varían según la cantidad de dimensiones que representa el estado x , que cantidad física se considera como entrada u y como esta entrada afecta el movimiento definido por F . Los modelos son simplemente aproximaciones de la física real y varían según la cantidad de supuestos que requieren.

2.1. Rigid Body Dynamics

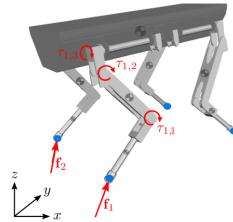


Figura 2: *Rigid Body Dynamics* (RBD).

Una posibilidad para modelar un sistema es a través de cada uno de sus cuerpos rígidos. Rígido en este contexto implica que los cuerpos no se deforman cuando se aplican fuerzas. Con esta suposición, el sistema de la figura 2 se puede describir completamente mediante las coordenadas generalizadas $q = [q_b^T q_j^T] \in \mathbb{SE}(3) \times \mathbb{R}^n$, que mantiene la posición y orientación de la base, así como el ángulo de cada articulación. Al ver un robot como una colección de cuerpos rígidos conectados a través de articulaciones accionadas, podemos derivar una relación entre el torque $\tau \in \mathbb{R}^n$ que actúa en cada articulación y el movimiento correspondiente como:

$$M(q)\ddot{q} + h(q, \dot{q}) = S^T\tau + J(q)^T f \quad (1)$$

donde $M \in \mathbb{R}(6+n) \times (6+n)$ es la matriz de inercia del espacio articular, $h \in \mathbb{R}(6+n)$ es el efecto de los términos centrifugo, de Coriolis y de gravedad, $S^T = [0n \times 6In \times n]^T$ es una matriz de selección que aplica el par de torsión τ solo a las n filas de articulaciones y el jacobiano J mapea las fuerzas $f = [f_1^T, \dots, f_{n_i}^T]^T \in R^{3n_i}$ en los n_i efectores finales a fuerzas generalizadas de dimensiones $6 + n$.

Se puede dividir la ecuación 2 en 6 filas desactivadas (u) y n activadas (a) [3]. Las filas desactivadas corresponden a las ecuaciones de la base y las otras filas describen los movimientos de las articulaciones.

$$Mu(q)\ddot{q} + hu(q, \dot{q}) = Ju(q)^T f \quad (2)$$

$$Ma(q)\dot{q} + ha(q, \dot{q}) = \tau + Ja(q)^T f \quad (3)$$

El subsistema desactivado (2) muestra matemáticamente que un torque τ no puede influir directamente en el movimiento de la base. Dicho de otra manera, no existe un motor para esta articulación del torso virtual de seis grados de libertad (Do). Los pies deben empujarse del suelo en la dirección opuesta para lograr un movimiento hacia adelante. Esta fuerza de contacto se puede generar a través de los motores en las articulaciones de las piernas, sin embargo, solo accionan indirectamente el torso o la base. La base se considera inactiva y hablamos de un sistema de base flotante. Este subsistema desactivado se puede utilizar para verificar que la relación entre las fuerzas externas f y las aceleraciones del sistema \ddot{q} sean físicamente correctas.

Una vez que se determinan estas dos cantidades, es trivial garantizar la relación definida por el subsistema accionado como muestra la ecuación 3. Dado que existe un motor generador de par para cada articulación del robot, este subsistema está completamente determinado. Al igual que para un manipulador de base fija, siempre es posible elegir pares apropiados τ para cumplir con estas ecuaciones accionadas.

Si los límites de par son lo suficientemente grandes, esto puede justificar que se ignore el subsistema accionado durante la planificación del movimiento. Al hacerlo, podemos eliminar n ecuaciones, así como la dependencia de los pares de torsión en las articulaciones en el problema de planificación, sin sacrificar la generalidad o la precisión. Podemos cambiar el énfasis a las seis filas que representan la dinámica no activada, como se hizo en [4] - [5].

Estas dinámicas cruciales para la locomoción de las piernas también se pueden expresar con respecto de masas instantáneo, a lo cual se le conoce como "Dinámica Centroidal".

2.2. Single Rigid Body Dynamics

Para que no exista una dependencia de los ángulos articulares, mientras se mantiene la dinámica aproximadamente correcta, son necesarios los siguientes supuestos adicionales:

- El momento producido por las velocidades articulares es insignificante.
- La inercia de todo el cuerpo sigue siendo similar a la de la posición articular nominal.

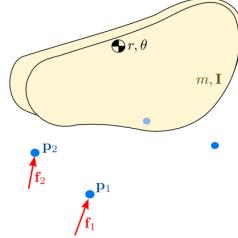


Figura 3: Single Rigid Body Dynamics (SRBD)

Esto podría suponerse razonablemente para robots en los que la base hace la mayor contribución a la masa total del robot. De la insignificante masa de las extremidades se deduce, en primer lugar, que incluso el movimiento rápido de las extremidades no contribuye significativamente al impulso del sistema y, en segundo lugar, que su posición no influye en la inercia de todo el cuerpo. Otro caso que podría justificar estas suposiciones es un robot que tiene masas importantes en las extremidades e inercia, pero que se mueven muy lentamente y se mantienen cerca de sus configuraciones nominales.

Al separar las partes lineales y angulares obtenemos las ecuaciones de Newton-Euler de un solo cuerpo rígido. La dinámica de un solo cuerpo rígido (SRBD) se define como:

$$m\ddot{r} = mg + \sum_{i=1}^{n_i} f_i \quad (4)$$

$$I(\theta)\dot{\omega} + \omega \times I(\theta)\omega = \sum_{i=1}^{n_i} f_i \times (r - p_i) \quad (5)$$

donde r es el centro de masas (CdM), θ es la orientación de la base y ω la velocidad angular de este cuerpo rígido combinado (ver Figura 3). La masa combinada de todas las extremidades y la base esta dada por m y $I(\theta) \in \mathbb{R}^{3*3}$ es la inercia de cuerpo completo de todas las extremidades individuales en la configuración de articulación nominal combinada, con respecto a un marco anclado en el CdM y cuyos ejes son paralelo al marco inercial.

A través de los supuestos adicionales, sacrificamos cierta precisión de la representación dinámica, sin embargo, obtenemos total independencia de los ángulos de las articulaciones. Esto nos permite expresar la dinámica puramente a través de coordenadas cartesianas y rotaciones en un espacio tridimensional lo que elimina la no linealidad introducida por los ángulos de

articulación. Además, en la locomoción con las piernas, a menudo es menos critico como se orientan exactamente las articulaciones de una pierna, siempre que el pie este en una posición especifica. Si descuidamos los efectos de los movimientos articulares sobre la dinámica, podemos desacoplar completamente la cinemática articular dependiente del robot del problema general de la locomoción física de las piernas. Esto permite una formulación mas simple de las leyes físicas y las limitaciones de la locomoción con las piernas y se ha utilizado en [6], [7].

Este modelo se encuentra en algún punto intermedio en cuanto a complejidad y precisión. Conserva el movimiento de la base de 6 dimensiones, así como las fuerzas de contacto individuales. Sin embargo, los productos cruzados aun introducen no linealidad. Esta no linealidad se elimina con otros supuestos como se ve a continuación.

2.3. Linear Inverted Pendulum Model

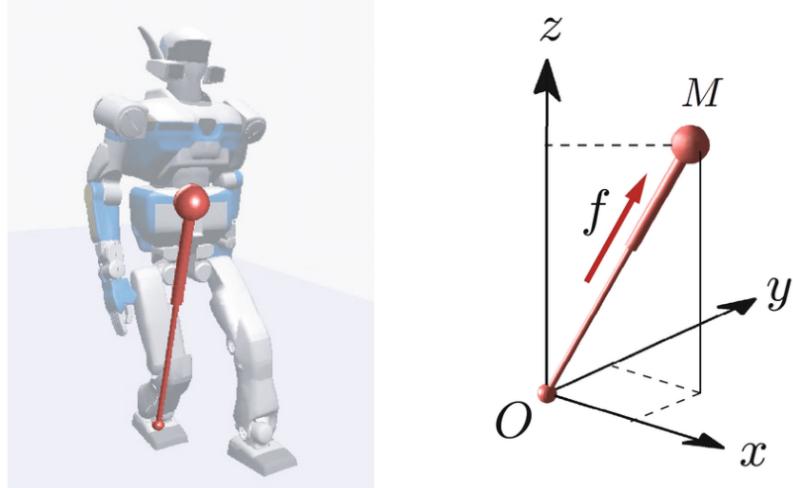


Figura 4: Linear Inverted Pendulum Model (LIPM)

Si partimos del modelo anterior, para eliminar la no linealidad introducida por el producto cruzado, son necesarios los siguientes supuestos adicionales: La altura de CdM r_z es constante. La velocidad angular ω y la aceleración $\dot{\omega}$ de la base son cero. Los puntos de apoyo están a una altura constante p_z .

Con estos supuestos, se simplifican las ecuaciones de forma que queda:

$$m\ddot{r}_x = \sum_{i=1}^{n_i} f_{i,x} = \frac{mg}{r_z - p_z} \left(r_x - \frac{\sum_{i=1}^{n_i} f_{i,z} p_{i,x}}{\sum_{i=1}^{n_i} f_{i,x}} \right) = \frac{mg}{h} (r_x - p_{c,x}) \quad (6)$$

donde x es el movimiento horizontal, h es la altura al caminar del robot y $p_{c,x}$ la posición x del centro de presión (CdP), que se manipula a través de las fuerzas de contacto verticales f_i , z_y la posición $p_{i,x}$ de los pies. Este modelo describe como la posición de la CdP afecta linealmente la aceleración horizontal de CdM. Se puede resolver de forma analítica y eficiente y, por tanto, se ha utilizado en una variedad de enfoques [8], [9] y [10].

Las desventajas de este modelo son las suposiciones restrictivas que se hicieron para que el modelo sea válido. Para movimientos más complejos, podría ser esencial reorientar el cuerpo para alcanzar puntos de apoyo específicos, acelerar verticalmente para saltar o colocar los pies a diferentes alturas para cruzar terrenos irregulares. Además, la entrada a este sistema es la CdP creada por una combinación de los componentes verticales de cada fuerza de contacto individual. Esta abstracción pierde información crítica sobre la fuerza vertical en cada pie, además de ignorar sus componentes tangenciales. Sin embargo, las fuerzas tridimensionales en cada pie son los elementos cruciales que mueven la base flotante y están sujetos a una variedad de restricciones (por ejemplo, cono de fricción). Al resumirlos en la CdP, se pierde información relevante que puede dificultar el modelado de muchas de las características de la locomoción con patas.

Se han desarrollado algunas extensiones para mitigar algunas de estas restricciones impuestas por la LIPM. Otra opción si el LIPM es demasiado restrictivo para un caso de uso específico (y se puede tolerar la no linealidad de la formulación) es usar directamente formulaciones más generales como el RBD completo o el SRBD. Estos siguen siendo solo modelos aproximados, pero se basan en menos suposiciones y, a veces, facilitan el reconocimiento de los principios físicos subyacentes. En el Cuadro 1 se puede ver un resumen de los diferentes modelos dinámicos descritos anteriormente y lo que caracteriza su estado y entrada.

Cuadro 1: Comparación de los distintos modelos dinámicos.

	Formula	Estado X	Entrada U
Rigid Body Dynamics	1, 2, 3	q_b, q_j	τ, f_i
Single Rigid Body Dynamics	4, 5	r, θ, p_i	f_i
Linear Inverted Pendulum Model	6	r_x, r_y	p_c

3. Física de la locomoción con patas

Lo que complica el control de los sistemas con patas en comparación con los rodantes o los voladores son las diversas restricciones sobre las fuerzas de reacción alcanzables. Estas restricciones de fuerza, que en consecuencia también limitan los posibles movimientos de la base, son:

1. Las fuerzas solo se pueden crear cuando un pie toca el medio ambiente. Esto restringe a que las fuerzas pueden actuar sobre superficie del terreno y en ningún otro lugar.
2. Físicamente, solo es posible crear fuerzas que empujen el suelo y no tiren de él. Las fuerzas unilaterales hacen que muchas aceleraciones de base sean físicamente imposibles.
3. Las fuerzas tangenciales, que son las que mueven el cuerpo hacia adelante, deben permanecer dentro del cono de fricción. Solo podemos crear fuerzas tangenciales más grandes aumentando simultáneamente la fuerza normal (fricción de Coulomb). Sin embargo, esto también afecta la aceleración del cuerpo en la dirección normal.

Estas fuertes restricciones y su efecto sobre el movimiento de la base es lo que hace que la locomoción con las piernas sea tan difícil desde el punto de vista de la planificación y el control.

4. Planificación de locomoción tradicional con patas

Previamente, se introdujeron los modelos dinámicos y las restricciones físicas que caracterizan la locomoción con patas. A continuación, se explicará brevemente los enfoques existentes para controlar robots con patas y discutir qué aspectos podrían mejorarse.

Para atravesar cualquier terreno, es útil saber donde pisar. Dado que la selección de puntos de apoyo es un problema difícil, a menudo se utiliza una variedad de heurísticas para simplificar dicha dificultad, por ejemplo clasificando los puntos de apoyo como buenos si están en un terreno plano para evitar resbalones. Asimismo, un mapa de altura del terreno suele ser la primera etapa de la planificación. Expresado en términos de las restricciones introducidas anteriormente, esto predefine donde (puntos de apoyo) y cuando (tiempos de paso) pueden actuar las fuerzas externas.

Solo después de que se hayan predeterminado estas ubicaciones de fuerza, se encuentra un movimiento corporal que puede ser creado con las fuerzas disponibles. Una vez que se han encontrado puntos de apoyo y un plan de movimiento base, el objetivo es aplicar esto a un sistema físico. Para ello, se requiere la generación de los pares de torsión apropiados para rastrear los movimientos de referencia.

Esta partición del problema en subtareas solucionables es favorable, pero también tiene sus limitaciones. Los puntos de apoyo especifican donde se permite aplicar las fuerzas de contacto. Posteriormente, se debe determinar un movimiento corporal físicamente correcto utilizando estas ubicaciones de contacto subestimas. Sin embargo, para movimientos complejos, puede resultar costoso leer estas posiciones sin tener en cuenta la dinámica de como las fuerzas generadas afectan a la base.

5. Optimización de Trayectorias

Imagina un satélite moviéndose entre dos planetas. Será usado el termino trayectoria para describir la ruta que toma el satélite entre los dos planetas. Por lo general, esta ruta incluirá tanto el estado (por ejemplo, la posición y la velocidad) como el control (por ejemplo, la fuerza del empuje) en función del tiempo. El termino optimización de trayectorias se refiere a un conjunto de métodos que se utilizan para encontrar la mejor elección de trayectoria, normalmente seleccionando las entradas al sistema (controles) como funciones en el tiempo.

Se dice que una solución a un problema de optimización de trayectorias es factible si satisface todas las requisitos del problema, es decir, las restricciones. En general, la optimización de trayectorias se ocupa de encontrar la mejor de las trayectorias factibles, que se conoce como la trayectoria óptima.

Como podemos comprobar, usando bucle abierto (optimización de trayectorias) encontramos que, a diferencia de usar bucle cerrado, hacemos uso de un método y solución local, es menos difícil de computar y es mas útil en problemas de dimensionalidad alta.

5.1. Métodos

En cuanto a la generación de movimientos para sistemas dinámicos sujetos a una variedad de restricciones físicas, el objetivo es reducir la cantidad de componentes manuales aprovechando las técnicas de optimización de trayectorias. Existen diferentes métodos para formular un problema de

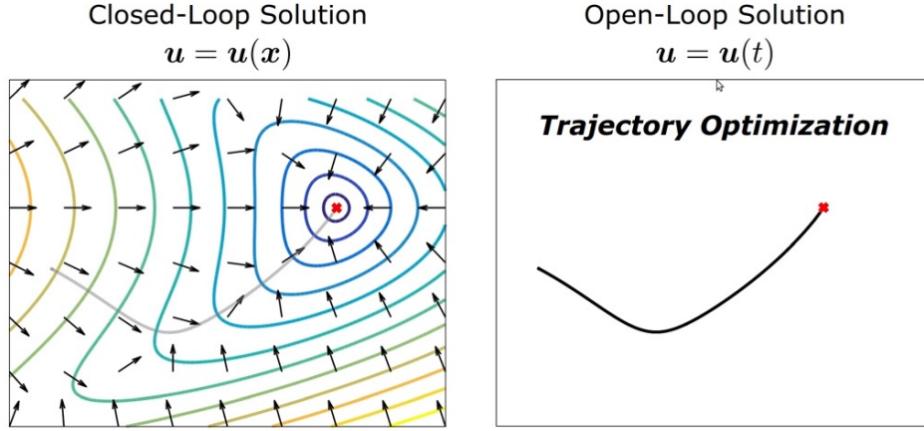


Figura 5: Comparación solución en bucle cerrado y en bucle abierto.

optimización de trayectorias, no obstante, este trabajo estará centrado en los métodos directos, es decir, en los que el problema de tiempo continuo de optimización de trayectorias se transforma en un problema de programación no lineal (PNL) afectado por un número finito de variables de decisión como,

$$\min_w a(w) \quad \text{sujeto a} \quad b(w) = 0, \quad c(w) \geq 0$$

Un solucionado intenta encontrar las variables w que minimizan el coste a , mientras se cumplan las restricciones de igualdad $b = 0$ y desigualdad $c \geq 0$. Después de formular el problema de esta manera, se puede resolver con una variedad de métodos para la optimización matemática. La ventaja de la optimización de trayectorias es que una vez que la física se ha modelado adecuadamente, el algoritmo puede producir movimientos para una amplia variedad de tareas, resolviendo la planificación de la locomoción a un nivel más general. Además, se deberá satisfacer la dinámica ODE (*Ordinary Differential Equation*). Para ello, se explican dos formas comunes de transformar un problema de optimización de trayectorias continuo a forma discreta dada (Ecuación 5.1).

5.1.1. Métodos secuenciales

Una forma intuitiva de transformar el problema de optimización de trayectorias continuo en un PNL de dimensión finita es mediante *single shooting*. Para ello, únicamente la entrada continua $u(t)$ se parametriza mediante un

conjunto de variables discretas. Esta entrada se aplica al sistema dinámico y los estados $x(t)$ se simulan hacia adelante usando un solucionador ODE. El solucionador de PNL verifica si los estados resultantes cumplen las restricciones de igualdad (b) y desigualdad (c) especificadas. Si este no es el caso, los parámetros w describiendo la trayectoria de entrada se adaptan en función de la información del gradiente y el sistema dinámico se simula (integrado hacia adelante) de nuevo. Este método 'dispara' continuamente la dinámica hasta encontrar parámetros de entrada para los cuales los estados evolucionan de acuerdo con el coste y las limitaciones.

Las ventajas de este método son que puede hacer uso de solucionadores ODE adaptables para el control de errores. Estos pueden cambiar su tamaño de paso de integración dependiendo de la dinámica actual, evitando así cálculos innecesarios y reduciendo el tiempo de calculo. La desventaja es que la trayectoria del estado no se puede inicializar directamente, ya que solo las variables de entrada intuitivas estarán disponibles.

5.1.2. Métodos simultáneos

Si las variables de decisión w parametrizan la entrada $u(t)$ y establecen $x(t)$ a lo largo del tiempo, el método se clasifica como método simultaneo. En este caso, la dinámica del sistema no es reforzado por la integración hacia adelante, sino a través de restricciones de igualdad en $b(w)$. El solucionador varía la trayectoria del estado $x(t)$ y la entrada $u(t)$ simultáneamente mientras se intenta hacer cumplir la relación de dinámica de sistemas entre ellos en momentos específicos. Esto requiere especificar los momentos en los que se aplica la restricción dinámica, y es computacionalmente costoso de cambiar durante las iteraciones si una precisión diferente es requerida. El lado positivo, es que un movimiento inicial se puede establecer fácilmente. También, debido al desacoplamiento entre el estado y la entrada, los cambios en la entrada solo afectan al estado en ese momento, mientras que los estados futuros no se verán afectados. Esta característica, que se refleja en el jacobiano, aumenta la robustez del solucionador hacia sistemas inestables. Debido a estas ventajas, para el caso del robot cuadrúpedo será usado variando en qué grado de libertad del sistema están optimizados, qué modelo dinámico se utiliza y cómo las restricciones de la locomoción están formuladas.

6. Algoritmos de planificación de movimiento

En esta sección compararemos diferentes enfoques capaces de producir movimientos de alta calidad.

Existe una variedad de trabajos exitosos que utilizan LIPM (*Linear Inverted Pendulum Model*) como modelo para generar movimiento planos [9]. En [11], [10] los autores optimizan el plan de movimiento con un MPC (*Model Predictive Control*), resultando en un controlador que puede reaccionar a las perturbaciones y manejar las imprecisiones del modelo. La selección del punto de apoyo y la generación del movimiento del cuerpo se desacopla para obtener esta velocidad. El enfoque presentado por [12] combina la selección del punto de apoyo plano y el movimiento del centro de masas lateral generado en el mismo problema de optimización en linea. La extensión no lineal de este trabajo, mientras sigue usando el LIPM, permite encontrar también las orientaciones óptimas para pies bípedos, evita obstáculos y sigue funcionando en linea en un HRP-2 [13]. Una optimización tan combinada de puntos de apoyo y movimiento del cuerpo también se muestra en [14] usando la colocación directa con un solucionador SQP (*Sequential Quadratic Programming*).

Para eliminar algunas de las restricciones impuestas por la LIPM, una variedad de trabajos exitosos para los robots humanoides se desarrollaron utilizando modelos como el CD (*Centroidal Dynamics*) o el RBD (*Rigid Body Dynamics*) completo. El punto de apoyo y la planificación del movimiento corporal a menudo se desacopla para hacer que el problema sea mas manejable. *Multiple shooting* se utiliza para generar y ejecutar el ascenso de escaleras utilizando un soporte de pasamanos en un bípedo físico [15]. El enfoque de [5] genera con éxito una variedad de movimientos para un robot bípedo. Esto se logra optimizando primero la trayectoria de la base de un robot y luego encontrando configuraciones de equilibrio estático, creando un algoritmo interactivo de planificación de múltiples contactos. Estas configuraciones se utilizan luego para generar movimientos-planes físicamente factibles utilizando un modelo CD.

De manera similar, también se pueden generar planos de movimiento físicamente correctos alternando entre encontrar el impulso del robot a partir de un plano de movimiento cinemático dado y luego optimizando las correspondientes fuerzas de contacto y ubicaciones para cumplir con el CD [16]. Un enfoque similar para separar el problema es optimizar alternativamente los puntos de apoyo y luego el movimiento del cuerpo y fuerzas coherentes con el CD [17]. Se muestran planos de movimiento para un robot humanoide completo, incluidas las articulaciones y demostrando como rastrearlas de manera eficiente.

Dado que predefinir el programa de contacto por adelantado puede restringir los movimientos alcanzables, se han desarrollado algoritmos que lo determinan automáticamente. Algunos enfoques [18] abordan esto fijando

solo la secuencia de contacto, pero permitiendo que la optimización eliminar fases específicas estableciendo su duración en cero. Otros codifican la noción dando un paso a través de variables binarias [19]. La secuencia de la marcha, junto con la CD, es resuelta usando *Mixed Integer Convex Programming* (MICP).

Existen otros métodos que dan al optimizador total libertad para elegir la programación de contactos y tiempos minimizando una función de coste. Los métodos de optimización como CMA-ES (Covariance Matrix Adaptation Evolution Strategy) se utilizan para determinar los parámetros del controlador que generan el movimiento con el mínimo coste. Si bien los planos de movimiento pueden ser bastante sofisticados, dicho estocástico y la técnica de optimización no basada en gradientes puede tardar en encontrar una solución. Otros procesos basados en el algoritmo SLQ (Sequential Linear Quadratic Programming) generan planos de movimientos para una amplia gama de sistemas lo suficientemente rápidos para ser utilizados en un bucle de control MPC. Este enfoque utiliza un modelo RBD completo y minimiza la función de coste utilizando un método Cuasi-Newton. Dado que estos enfoques se basan principalmente en una función de coste, en lugar de restricciones estrictas, pueden requerir un ajuste cuidadoso de la ponderación parámetros para lograr la convergencia de tareas más complejas.

Otra forma de representar el problema de la locomoción es mediante una combinación de costes y restricciones duras de LCP (*Linear Complementary Problem*) que modelan el contacto con el entorno. Los enfoques exitosos generan planos de movimiento dinámicos que optimizan también la marcha y tiempos de paso utilizando el CD o el RBD completo [4]. Incluyen directamente límites conjuntos y otras restricciones en el problema de optimización y muestran una variedad de movimientos, incluyendo fases de vuelo. Desde un punto de vista mas critico, este modelado del contacto a través de un LCP puede ser costoso computacionalmente, lo que podría mitigarse mediante nuestra parametrización del efecto final. Si bien esto permite optimizar la secuencia de pasos de manera eficiente, también requiere fijar el número de pasos por adelantado, lo cual los algoritmos anteriores pueden descubrir automáticamente.

Cada uno de estos enfoques tiene sus ventajas y desventajas. Para comparar diferentes algoritmos de planificación de movimiento, es útil evaluarlos en función de:

- Que componentes están optimizados.
- Que tan difíciles son las tareas.

- Que tan fiables son las soluciones encontradas.
- Cuanto tiempo lleva encontrar una solución.
- Si se han verificado las soluciones en simulación y en un robot físico.

7. Ejemplo de aplicación a robots cuadrúpedos

Los robots con patas son sistemas híbridos (su dinámica cambia de forma brusca dependiendo de los contactos que realiza con su entorno) e infraac-tuados con dinámicas altamente no lineales. Esto significa que realizar su control es una tarea muy difícil.

Una vez descrita la Optimización de Trayectorias como método basado en el control óptimo para encontrar soluciones únicas a problemas de control específicos, se procede a explicar su aplicación particular en el caso de robots móviles con patas. Mas concretamente, se va a detallar su uso tal como esta descrito en [7], donde se generan trayectorias para el robot HyQ [20] y ANYmal [21]. *towr* es el nombre de este proyecto y se puede encontrar su código en *GitHub*. Se trata de una librería preparada para trabajar con ROS (aunque también se puede usar de forma independiente), que facilita la interacción con un optimizador no lineal para tareas relacionadas con la robótica.

En este trabajo se presenta una formulación que permite la generación de trayectorias que automáticamente determina la secuencia del andar (en que momento y en qué orden hacer las pisadas), las trayectorias de cada pie en el aire y el movimiento del cuerpo del robot en seis dimensiones, todo esto sobre terrenos que no son planos. Sin embargo, una limitación es que hay que indicar el numero de pasos con antelación.

Ya que el cuerpo del robot va a poder moverse libremente en sus seis grados de libertad (3 de traslación y 3 rotacionales), no se va a asumir el modelo dinámico del péndulo invertido lineal inverso como en otras muchas aplicaciones, ya que no es lo suficientemente rico como para expresar en su simplificación las dinámicas de un sólido rígido. En su lugar, se va a usar el modelo SRGB (por sus siglas en inglés, *Single Rigid Body Dynamics*), que si que es capaz de modelar mas fielmente la dinámica el robot, asumiendo que sus piernas tienen un peso despreciable con respecto al cuerpo.

Las trayectorias se generan con un optimizador no lineal (NLP, por sus siglas en inglés, *Non Linear Optimizer*). Existen en el mercado infinidad de optimizadores, no habiendo ninguno mejor que otro y todos funcionando

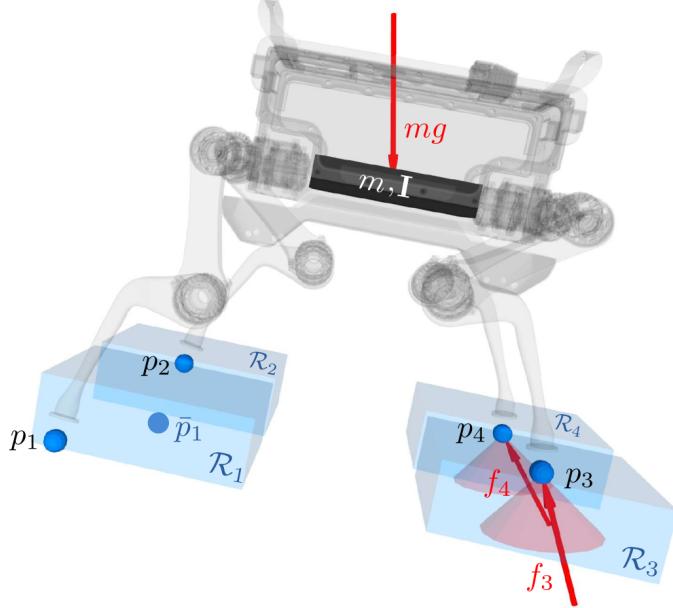


Figura 6: Esquema del modelo del robot que ve el optimizador

mejor para ciertas tareas en específico. Para este caso se hace uso del optimizador IPOPT [22], un optimizador gratuito y de código abierto con una licencia permisiva.

7.1. Formulación

Para poder hacer uso del optimizador no lineal, tenemos que definir el problema como un problema de optimización, esto es, encontrar los parámetros que minimizan nuestro coste y cumplen con las restricciones que se piden.

En este caso en particular, no ha hecho falta especificarle un coste al optimizador, pues con que se cumplan las restricciones (descritas en el Apartado 7.1.2), ya se considera que se encuentra una trayectoria lo suficientemente realista como para pasársela a un controlador.

7.1.1. Parametrización

El objetivo es obtener las posiciones y orientaciones del centro de masas del cuerpo del robot, así como la posición de sus patas. Como conocimiento

previo, se conocen el numero de pasos que el robot tiene que dar y el tiempo total en el que tiene que darlos, T . Sin embargo, no hace falta especificar la duración de cada paso, ya que se parametrizan las duraciones como valores continuos $\Delta T_{i,j}$ que el optimizador tiene que encontrar, dotándolo de mas libertad para buscar un rango mas amplio de posibles movimientos (y por tanto, ralentizándolo).

Para parametrizar las trayectorias en el espacio del cuerpo del robot y las patas, se podría optar por guardar los valores de cada variable para cada instante de tiempo. Pero esto incrementaría de manera lineal con el tiempo el numero de variables que el optimizador tendría que optimizar. Una forma de reducir este crecimiento innecesario en el numero de variables es asumir que para intervalos pequeños de tiempo las trayectorias van a ser continuas.

De esta forma, se pueden modelar los pequeños instantes de tiempo con los pocos parámetros que definen un *spline*. Para la posición y orientación del cuerpo del robot se usa una secuencia de *splines* de cuarto orden de manera que abarque la totalidad del tiempo T . De manera parecida, las trayectorias de cada componente de la posición de las patas en $\Delta T_{i,j}$ se modelan como otra secuencia de tres *splines* de grado tres, ya que se asume que las patas no necesitan mayor libertad de movimiento que la descrita por polinomios de tercer grado.

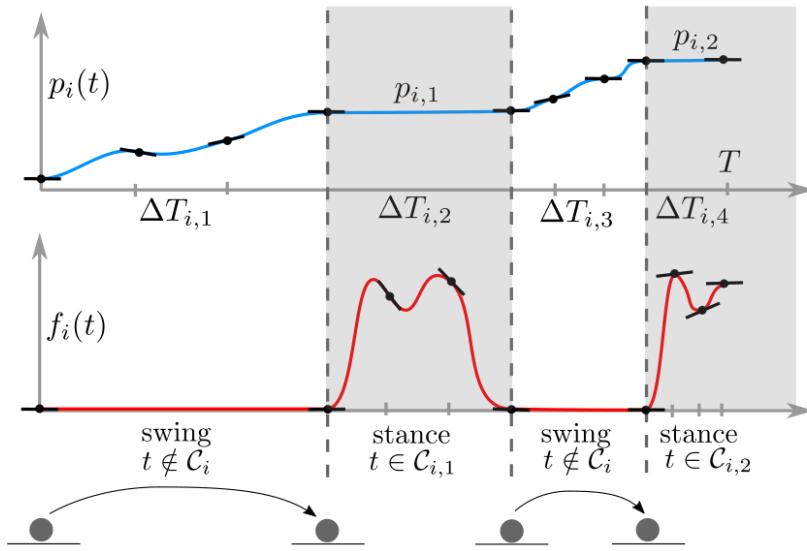


Figura 7: Esquema grafico de la parametrizacion del movimiento de cada pata, descrita en el Apartado 7.1.1.

Como las variables resultantes de la optimización describen tanto la entrada (señales de control que se buscan) como la evolución del robot (su movimiento siguiendo las físicas reales), se puede considerar que este método como uno de colocación.

7.1.2. Restricciones

Además de especificarle al optimizador las soluciones que tiene que buscar, hay que indicarle cuales son las combinaciones de variables que no se consideran como solución, es decir, las restricciones.

Esto es necesario porque en el mundo real, el robot no es capaz de volar, plegarse sobre si mismo o atravesar el suelo, por ejemplo.

A continuación se explican las restricciones necesarias para la generación de estos movimientos.

1. Modelo cinemático

En otras formulaciones mas complejas donde se trabaja desde el espacio articular del robot, se limitan directamente los ángulos a los que puede girar cada una de ellas. En este caso, lo que se hace es limitar cada pata del robot a un espacio de trabajo dentro de un ortoedro (los que se pueden ver en azul en la Figura 6). Asegurándose previamente que las patas puedan moverse libremente dentro de su respectivo ortoedro, se consigue limitar el rango de movimiento cinemático del robot, evitando así configuraciones cinemáticas imposibles, de una manera mucho mas sencilla y amigable para el optimizador, lo cual acelerara el proceso de búsqueda de una posible solución.

Además, al ser las trayectorias continuas, la condición de que las patas se mantengan dentro de sus respectivas áreas de trabajo se realiza de forma periódica cada cierto intervalo pequeño de tiempo en vez de siempre, para facilitar la tarea al optimizador mas aun.

2. Modelo dinámico

Como se ha comentado antes, se pretende usar la dinámica de SRBD en vez de modelos mas sencillos, como el péndulo lineal invertido, debido a su baja expresividad que no permite realizar movimientos mas complejos y la navegación sobre terrenos abruptos.

Se podría trabajar también con la dinámica del sólido rígido expresada en el espacio de las articulaciones, pero las restricciones son mas intuitivas y sencillas de expresar con respecto a la dinámica cartesiana, por lo que se usa esta ultima.

En concreto, la dinámica del robot se expresa como:

$$m\ddot{\mathbf{r}}(t) = \sum_{i=1}^{n_i} \mathbf{f}_i(t) - m\mathbf{g} \quad (7)$$

$$\mathbf{I}\dot{\omega}(t) + \omega(t) \times \mathbf{I}\omega(t) = \sum_{i=1}^{n_i} \mathbf{f}_i(t) \times (\mathbf{r}(t) - \mathbf{p}_i(t)) \quad (8)$$

Donde m es la masa del robot, n_i es el numero de patas, \mathbf{g} es la aceleración de la gravedad y $\omega(t)$ es la velocidad angular que puede ser calculada a partir de los ángulos de Euler resultantes de la optimización $\theta(t)$ y $\dot{\theta}(t)$.

La inercia \mathbf{I} se calcula con el robot en su posición nominal, y es constante aunque las piernas cambien de posición, ya que se asume que tienen un peso despreciable.

La restricción consiste en hacer que 7 se cumpla cada ciertos intervalos de tiempo a lo largo de toda la trayectoria. Además, se pone una restricción mas para asegurarse de que las aceleraciones en la unión de dos *splines* sean iguales, de esta forma se evitan saltos en la aceleración indeseados (ya que estos saltos implicarían discontinuidades en las fuerzas externas o la posición de los pies).

El modelo dinámico descrito hasta ahora perfectamente puede servir para describir los movimientos de un *quadrotor*. Lo que lo hace específico para el caso de robots móviles con patas son las restricciones de las fuerzas y posiciones de las patas, que cambian abruptamente cuando cada pata entra en contacto con el suelo o despega. Estas discontinuidades no trabajan de forma natural con la forma de operar de los optimizadores no lineales.

Para simplificar un poco este problema, se puede tratar cada pata del robot de manera individual, teniendo en cuenta las fuerzas de las patas que están en el suelo. Esto hace que se puedan modelar los contactos de cada pata con una variable *booleana*, y que la misma formulación funcione para robots con distinto numero de patas.

3. Modelo de los contactos Para un robot bípedo, todas las combinaciones de los contactos posibles en un instante son cuatro: (L, R, D, F) , como se pueden ver en la Figura 8. En casos sencillos como estos, es

posible modelar cada una de las combinaciones de los contactos. Pero para robots con mas patas o puntos de contacto como por ejemplo cuadrúpedos, hexápodos o bípedos que hacen uso de sus brazos en el entorno pueden llegar a tener un numero muy alto de combinaciones de contactos y por tanto de modelos dinámicos distintos.

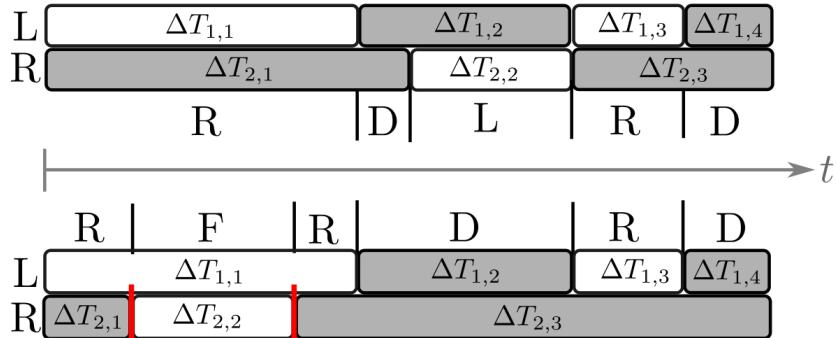


Figura 8: Dos programaciones de contactos distintas para robots bipedos. L y R hacen referencia a los pies izquierdo y derechos respectivamente.

Como se ha comentado anteriormente, si se conocen el numero de pasos a dar previamente y se considera cada pie como independiente, se pueden optimizar los valores de tiempo $\Delta T_{i,j}$ ya que son continuos. Esto hace que se puedan usar optimizadores no lineales directamente sin tener que realizar Programación en Enteros.

Por tanto, el optimizador puede decidir que secuencia de pasos llevar a cabo, simplemente cambiando los valores de duración de cada paso individualmente. Sin embargo, la suma de las duraciones de todos los pasos de cada pie tiene que ser igual al tiempo total T , por lo que se tiene que especificar una restricción mas para capturar esto.

Por ultimo, si un pie del robot no esta en contacto con el suelo, su fuerza externa por lógica tiene que ser nula. Esto también se necesita especificar con una restricción mas.

4. Altura del terreno Cuando los pies del robot no están en el aire, tienen que estar sobre el terreno. En caso contrario, los pies tienen que estar por encima del mismo y no debajo del suelo.

La formulación permite usar suelos representados mediante mapas de altura, por lo que el robot puede atravesar terrenos complicados como

el que se puede ver en la Figura 9.

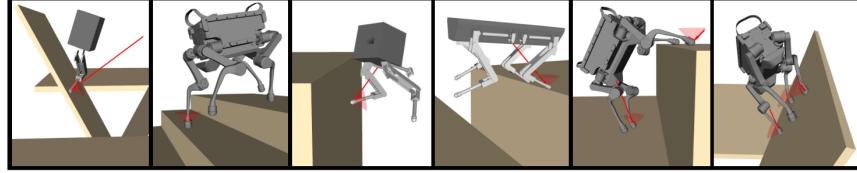


Figura 9: Ejemplos de robots siguiendo las trayectorias generadas por el optimizador en terrenos complicados.

5. Fuerzas en reposo

Para optimizar de forma correcta las físicas del robot, se tiene que tener en cuenta que las fuerzas de reacción que el robot puede generar son de repulsión contra el suelo, ya que las patas del robot no se pueden agarrar al suelo para tirar en vez de empujar. Por ello, es necesario especificarle al optimizador en forma de restricción de desigualdad que esto no es posible, asegurándonos de que las fuerzas obtenidas sean siempre de empuje (como las fuerzas f_3 y f_4 de la Figura 6).

7.1.3. Formulación matemática

Una vez explicados todos los puntos anteriores, en esta sección se muestra la formulación de la optimización de trayectorias aplicado a este problema de forma matemática:

encuentra $\mathbf{r}(t) \in R^3$ (Posicion del Centro de Masas)
 $\boldsymbol{\theta}(t) \in \mathbb{R}^3$ (Orientacion de la base)

para cada pie i :

$\Delta T_{i,1}, \dots, \Delta T_{i,2n_{s,i}} \in \mathbb{R}$ (Duracion de las fases)
 $\mathbf{p}_i(t, \Delta T_{i,1}, \dots) \in \mathbb{R}^3$ (Posiciones de las patas)
 $\mathbf{f}_i(t, \Delta T_{i,1}, \dots) \in \mathbb{R}^3$ (Fuerza de los pies)

tal que: $[\mathbf{r}, \boldsymbol{\theta}](t=0) = [\mathbf{r}_0, \boldsymbol{\theta}_0]$ (Estado inicial)
 $\mathbf{r}(t=T) = \mathbf{r}_g$ (Estado final)
 $[\ddot{\mathbf{r}}, \dot{\boldsymbol{\omega}}]^T = \mathbf{F}(\mathbf{r}, \mathbf{p}_1, \dots, \mathbf{f}_1, \dots)$ (Modelo dinamico)

para cada pie i :

$\mathbf{p}_i(t) \in \mathcal{R}_i(\mathbf{r}, \boldsymbol{\theta})$ (Modelo cinematico)

si el pie i esta en contacto:

$\dot{\mathbf{p}}_i(t \in \mathcal{C}_i) = \mathbf{0}$ (Sin deslices)
 $p_i^z(t \in \mathcal{C}_i) = h_{terreno}(\mathbf{p}_i^{xy})$ Altura del terreno
 $\mathbf{f}_i(t \in \mathcal{C}_i) \cdot \mathbf{n}(\mathbf{p}_i^{xy}) \geq 0$ (Fuerza de empuje)
 $\mathbf{f}_i(t \in \mathcal{C}_i) \in \mathcal{F}(\mu, \mathbf{n}, \mathbf{p}_i^{xy})$ (Fuerza dentro del cono de friccion)

si el pie i esta en el aire:

$\mathbf{f}_i(t \in \mathcal{C}_i) = \mathbf{0}$ (no hay fuerza estando en el aire)

$\sum_{j=1}^{2n_{s,i}} \Delta T_{i,j} = T$ (Duracion total)

7.2. Resultados

Los resultados se pueden ver en el video adjunto a esta entrega. Los movimientos mostrados son simplemente las trayectorias encontradas siendo mostradas, no están siendo simuladas por un motor de físicas como podría ser Gazebo. El optimizador solo calcula las posiciones de la base y de los efectores finales del robot, por lo que es necesario implementar la cinemática inversa de todas las patas para calcular los ángulos de cada articulación y poder mostrarlos.

Para poder ejecutar esta demostración, es necesario instalar los siguientes paquetes presentes en los repositorios de ROS Kinetic:

- **xpp**: `sudo apt install ros-kinetic-xpp`

- **ifopt**: `sudo apt install ros-kinetic-ifopt`

Mientras que los siguientes paquetes se tienen que clonar en un *workspace*, ya que son los modificados por los alumnos para incluir el robot Talos:

- **towr** (modificado): clonar el siguiente repositorio, <https://github.com/noctrog/towr/tree/talos>
- **xpp_talos** (visualización y cinemática inversa del robot Talos): https://github.com/noctrog/xpp_talos

Para compilarlos, se tiene que usar el comando

```
catkin build --force-cmake -DCATKIN_BUILD_TYPE=Release
```

Para ejecutarlo, se tienen que seguir los pasos indicados en el vídeo adjunto en la portada.

Se adjunta un vídeo de los autores donde se muestran varios ejemplos de trayectorias en distintos robots y entornos¹.

Para poder ejecutar estas trayectorias en un robot real, es necesario programar un controlador que sea capaz de replicar las fuerzas de contacto generadas mientras se sigue la trayectoria. Esto es un área de igual magnitud de trabajo que la tarea de generar las trayectorias, por lo que se sale del propósito de este trabajo. Aun así, cabe mencionar que se requiere usar controladores que resuelven un programa cuadrático constantemente para actuar de la forma mas óptima posible dadas unas restricciones [11] (este es el controlador que se usa al final del vídeo anterior).

Para comprobar la correcta optimización de las físicas del robot, componente critico de la optimización de trayectorias, se muestra en la Figura 10 la relación entre la aceleración real y la optimizada.

8. Ejemplo aplicado a robots bípedos

En este apartado se va a mostrar de forma rápida como podría ser una formulación más focalizada al problema de la locomoción de bípedos, ya que como se ha podido ver en la vídeo demostración, *towr* no ha dado buenos resultados al estar originalmente pensado para robot cuadrúpedos.

En [23] realizan optimización de trayectorias para el robot bípedo *Cassie* (Figura 11). Para ello, usan C-FROST, un *frontend* del paquete FROST para

¹Link del vídeo: <https://youtu.be/0jE46GqzxMM>.

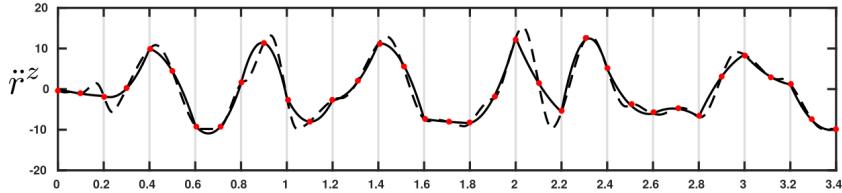


Figura 10: Comparación de la aceleración vertical del cuerpo del robot real (en linea discontinua) y la optimizada (linea continua). Se puede apreciar que la aceleración optimizada solo es igual a la real en los puntos en los que hay restricciones (puntos rojos). Entre ellos la aceleración se desvía ligeramente, pero esto no tiene mucha importancia ya que aunque las aceleraciones fueran perfectas el controlador no podría seguirlas de manera exacta.

MATLAB. Su objetivo es convertir formulaciones escritas originalmente en MATLAB a C++ nativo, usando como optimizador IPOPT (el mismo que el usado en el Apartado 7).

Al ser un método basado en colocación también, tiene muchas cosas en común con el ejemplo anterior por lo que no se va a volver a explicar. Las diferencias más notables son que en este caso sí que se tiene en cuenta la dinámica total del robot (*Rigid Body Dynamics*) y las restricciones cambian. Esto hace que para encontrar una trayectoria, se pase de unos pocos segundos del ejemplo anterior a aproximadamente unos diez minutos para el robot *Cassie*.

Las restricciones de la formulación para el robot *Cassie* son:

- Los pasos tienen una duración fija de 0.4s, por lo que el optimizador no puede elegir arbitrariamente cuanto tiempo dura cada paso como en *towr*.
- Los pies están a una altura mínima de 15cm cuando están a mitad del paso.
- Las fuerzas de reacción con el suelo tienen que mantenerse dentro del cono de rozamiento (esta restricción no cambia).
- Se introduce el concepto del *Zero Moment Point* y se asegura en todo momento que tiene que estar dentro del polígono convexo formado por los contactos de los pies con el suelo en cada instante [24].
- Velocidad horizontal nula a la hora de aterrizar el pie.

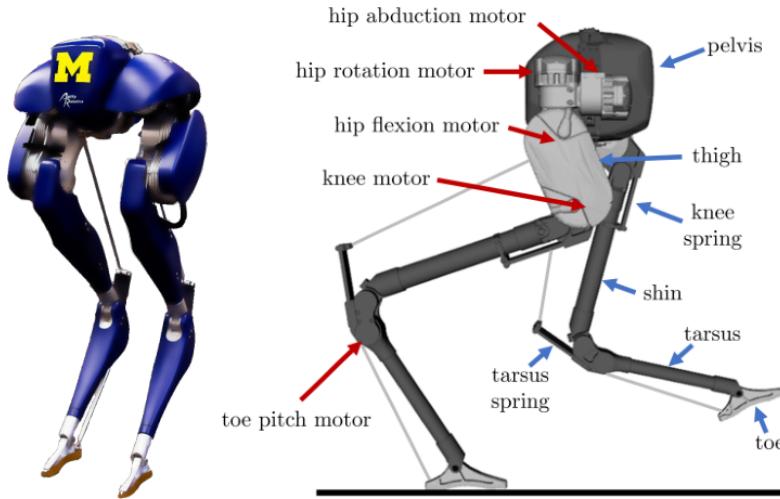


Figura 11: El robot *Cassie*.

- Para un andar en el plano sagital, los dos pasos del andar tienen que ser simétricos.
- Los andares sagitales tienen límites más estrechos en la abducción de la cadera y motores rotacionales que otros andares que permiten movimiento lateral.
- El torso tiene que mantenerse entre ± 3 grados en *pitch* y *roll*.

9. Optimización de Trayectorias en el futuro

A largo plazo, el objetivo más claro de este paradigma de control es llegar a poder ejecutarlo en bucle cerrado. Ya que al ser capaces de correrlo en bucle cerrado, se hace posible que los robots reaccionen a cambios en el entorno o a grandes perturbaciones externas como pueden ser colisiones o empujones.

Pero para poder hacer posible que se realice la optimización en tiempo real, en general hay que hacer una serie de asunciones que permiten al optimizador funcionar de manera rápida. Pero éstas limitan la capacidad de movimiento del robot, ya que se suelen asumir dinámicas simplificadas (como se ha visto hasta ahora) o terrenos sencillos. Aún así, hay casos en los que se han conseguido desarrollar controladores sorprendentemente robustos sobre

esta idea. Uno de ellos es el desarrollado por el MIT para el robot Cheetah [6].

Sin embargo, está surgiendo recientemente una nueva ola de investigación que pretende juntar las áreas del aprendizaje automático con la de teoría de control. El objetivo es conseguir la expresividad y generalización del aprendizaje automático moderno, con el rigor y la formalidad de la teoría de control.

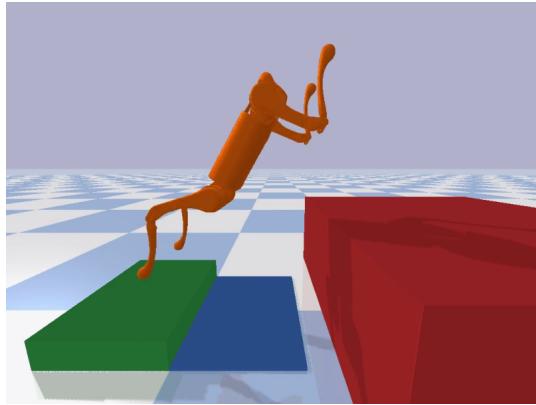


Figura 12: El robot cuadrúpedo *LASER mini*

Uno de estos ejemplos se encuentra en este reciente trabajo [25]. En él, se consigue optimizar la trayectoria para que un robot cuadrúpedo realice un salto (Figura 12). Sin embargo, si se hace la optimización de la trayectoria y se usan directamente las referencias como entrada de los controladores PD de las patas, no se consigue reproducir la trayectoria fielmente en el simulador y el robot se cae. Para evitar esto, se hace uso de *Deep Reinforcement Learning* para que el robot aprenda una política que sirva de apoyo a los controladores PD tanto del espacio de las articulaciones del espacio cartesiano (Figura 13). A esta política se entrena con el algoritmo *off-policy* conocido como SAC, por sus siglas en inglés (*Soft Actor Critic*).

Uno de los avances más importantes, además del más reciente, en este campo es el conseguido por los investigadores Jonhoo Lee y Hwangbo [26]. En este trabajo, se consigue un controlador basado totalmente en *Deep Reinforcement Learning*, obviando por completo la optimización de trayectorias. Para obtener este controlador, ha sido necesario la creación de un nuevo motor de físicas llamado *Raisim* [27]. *Raisim* es capaz de simular miles de robots *ANYmal* [21] en tiempo real y con una gran fidelidad. Esto ha permitido el entrenamiento del controlador basado en DRL con ingentes cantidades de

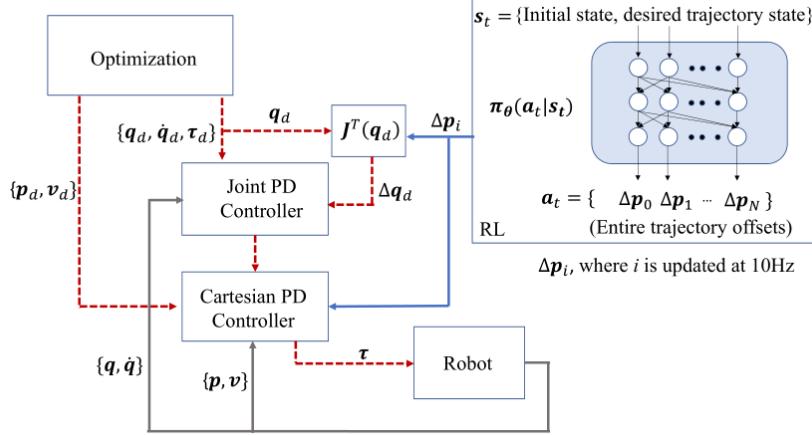


Figura 13: Diagrama de control que integra el agente representado por una red neuronal y los controladores tradicionales. Se puede ver que desde arriba se generan las trayectorias deseadas, que pasan a los controladores. Mientras que por otro lado la red neuronal genera todas las Δp_0 y las va pasando ordenadamente a los controladores de manera secuencial a una frecuencia de 10Hz.

datos.

Pero para poder realizar el entrenamiento desde cero, primero se tiene que partir de entornos sencillos. Es por eso que mediante un filtro de partículas se va seleccionando la dificultad del terreno por el que tiene que moverse el robot, incrementándola o reduciéndola según cómo de bien o mal lo esté haciendo el controlador.

El controlador resultante puede ser aplicado directamente a un robot real, sin tener que realizar ningún ajuste (algo muy normal en algoritmos que son probados en simulación). Además, el robot es capaz de atravesar cualquier terreno, por muy abrupto o resbaladizo que sea. Se realizó una prueba llevando al robot por varios terrenos (montañoso, nevado, en la orilla de un río) de una sola vez y no se cayó en ningún momento (Figura 14).

Pero lo más sorprendente es que el robot no está haciendo uso de ningún sensor exteroceptivo, simplemente crea una representación interna de lo que piensa que tiene en su entorno inmediato sólo con los contactos de sus piernas con el suelo.

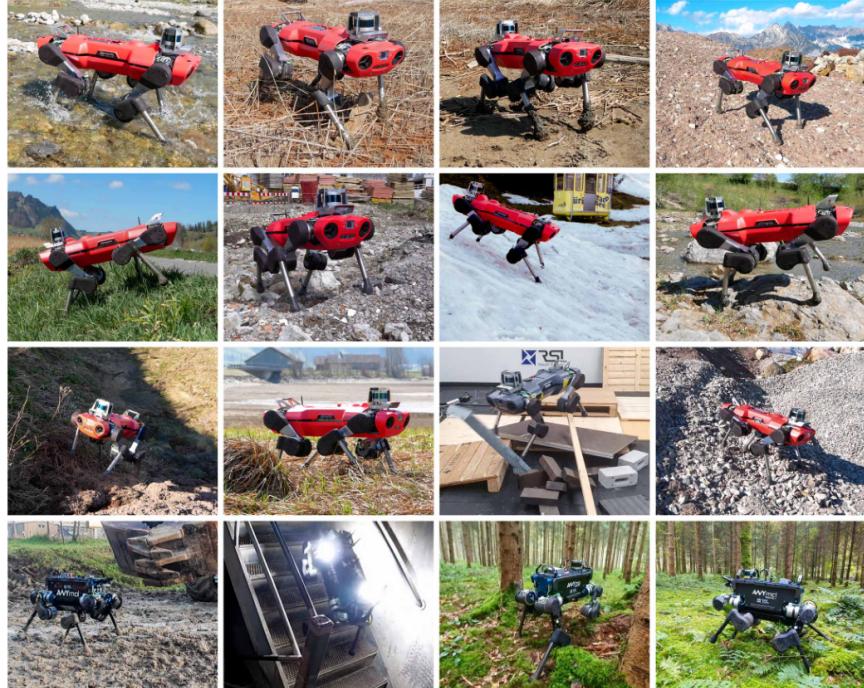


Figura 14: Prueba en entornos reales.

Se puede pensar que el futuro de la locomoción de robots con patas puede ir en esta prometedora dirección, y la optimización de trayectorias quede relegada a realizar el planificado de alto nivel.

Referencias

- [1] L. Righetti, J. Buchli, M. Mistry, and S. Schaal, “Inverse dynamics control of floating-base robots with external constraints: A unified view,” in *2011 IEEE International Conference on Robotics and Automation*, pp. 1085–1090, 2011.
- [2] F. Farshidian, E. Jelavić, A. Winkler, and J. Buchli, “Robust whole-body motion control of legged robots,” 03 2017.
- [3] D. Pardo, M. Neunert, A. Winkler, R. Grandia, and J. Buchli, “Hybrid direct collocation and control in the constraint-consistent subspace for dynamic legged robot locomotion,” 07 2017.

- [4] H. Dai, A. Valenzuela, and R. Tedrake, “Whole-body motion planning with centroidal dynamics and full kinematics,” in *2014 IEEE-RAS International Conference on Humanoid Robots*, pp. 295–302, 2014.
- [5] J. Carpentier, S. Tonneau, M. Naveau, O. Stasse, and N. Mansard, “A versatile and efficient pattern generator for generalized legged locomotion,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3555–3561, 2016.
- [6] G. Bledt, P. M. Wensing, and S. Kim, “Policy-regularized model predictive control to stabilize diverse quadrupedal gaits for the mit cheetah,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4102–4109, 2017.
- [7] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, “Gait and trajectory optimization for legged systems through phase-based end-effector parameterization,” *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.
- [8] A. W. Winkler, C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell, and C. Semini, “Planning and execution of dynamic whole-body locomotion for a hydraulic quadruped on challenging terrain,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5148–5154, 2015.
- [9] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, “Biped walking pattern generation by using preview control of zero-moment point,” in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 2, pp. 1620–1626 vol.2, 2003.
- [10] C. Dario Bellicoso, F. Jenelten, P. Fankhauser, C. Gehring, J. Hwangbo, and M. Hutter, “Dynamic locomotion and whole-body control for quadrupedal robots,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3359–3365, 2017.
- [11] C. D. Bellicoso, F. Jenelten, C. Gehring, and M. Hutter, “Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots,” *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2261–2268, 2018.

- [12] A. Herdt, H. Diedam, P.-B. Wieber, D. Dimitrov, K. Mombaur, and M. Diehl, “Online walking motion generation with automatic foot step placement,” *Advanced Robotics*, vol. 24, pp. 719–737, 04 2010.
- [13] K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, and T. Itozumi, “Humanoid robot hrp-2,” in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, vol. 2, pp. 1083–1090 Vol.2, 2004.
- [14] C. Gehring, C. Dario Bellicoso, P. Fankhauser, S. Coros, and M. Hutter, “Quadrupedal locomotion using trajectory optimization and hierarchical whole body control,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4788–4794, 2017.
- [15] M. Kudruss, M. Naveau, O. Stasse, N. Mansard, C. Kirches, P. Soueres, and K. Mombaur, “Optimal control for whole-body motion generation using center-of-mass dynamics for predefined multi-contact configurations,” in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pp. 684–689, 2015.
- [16] A. Herzog, S. Schaal, and L. Righetti, “Structured contact force optimization for kino-dynamic motion generation,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2703–2710, 2016.
- [17] B. Ponton, A. Herzog, S. Schaal, and L. Righetti, “A convex model of momentum dynamics for multi-contact motion generation,” 2016.
- [18] C. Mastalli, M. Focchi, I. Havoutis, A. Radulescu, S. Calinon, J. Buchli, D. G. Caldwell, and C. Semini, “Trajectory and foothold optimization using low-dimensional models for rough terrain locomotion,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1096–1103, 2017.
- [19] B. Aceituno-Cabezas, C. Mastalli, H. Dai, M. Focchi, A. Radulescu, D. G. Caldwell, J. Cappelletto, J. C. Grieco, G. Fernandez-Lopez, and C. Semini, “Simultaneous contact, gait and motion planning for robust multi-legged locomotion via mixed-integer convex optimization,” *IEEE Robotics and Automation Letters*, p. 1–1, 2017.
- [20] C. Semini, N. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. Caldwell, “Design of hyq -a hydraulically and electrically actuated

quadruped robot,” *Proceedings of the Institution of Mechanical Engineers. Part I: Journal of Systems and Control Engineering*, vol. 225, pp. 831–849, 08 2011.

- [21] M. Hutter, C. Gehring, D. Jud, A. Lauber, D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, R. Diethelm, S. Bachmann, A. Melzer, and M. Höpflinger, “Anymal - a highly mobile and dynamic quadrupedal robot,” *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 38–44, 2016.
- [22] A. Wächter and L. Biegler, “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming,” *Mathematical programming*, vol. 106, pp. 25–57, 03 2006.
- [23] A. Hereid, O. Harib, R. Hartley, Y. Gong, and J. W. Grizzle, “Rapid trajectory optimization using c-frost with illustration on a cassie-series dynamic walking biped,” 2019.
- [24] J. W. Grizzle, C. Chevallereau, R. W. Sinnet, and A. D. Ames, “Models, feedback control, and open problems of 3d bipedal robotic walking,” *Automatica*, vol. 50, no. 8, pp. 1955 – 1988, 2014.
- [25] G. Bellegarda and Q. Nguyen, “Robust quadruped jumping via deep reinforcement learning,” 2020.
- [26] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, “Learning quadrupedal locomotion over challenging terrain,” *Science Robotics*, vol. 5, p. eabc5986, Oct 2020.
- [27] J. Hwangbo, J. Lee, and M. Hutter, “Per-contact iteration method for solving contact dynamics,” *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 895–902, 2018.