第二十四屆旺宏金矽獎

作品企劃書

智慧雙足機器人應用於多層倉儲之人機協作

Human-Robot Collaboration of Intelligent Bipedal Robots in Multi-Level Warehousing

參賽組別：應用組

參賽編號：A24-073

隊　　長：卓冠廷

隊　　員：李照棋、黃瑋翔、陳彥銘

摘要

本企劃設計及應用身長 126.4 公分之雙足機器人，著眼於下半身機構應用，加入機構設計的巧思並開發系統架構，結合多種感測器資訊去對雙足機器人進行控制。

在設計方面，團隊根據人體肌群，以四連桿傳動方式對機器人重心做調整，不僅節省能源消耗，更是簡化機器人控制層面的難易度，同時也參考人體足弓，設計了具緩震效果的腳掌，降低馬達與機構損耗，克服地形限制，更是讓雙足機器人動作切合實體人類步態模式。

在應用方面，首先得建立雙足機器人步態數學模型，團隊成功將其運用於實體機器人上，接著透過三種不同的場域測試進一步驗證，有負重走路、上下階梯與不平坦地形的實驗，主打搬運機器人於多層倉儲會遇到的情境去研發，在基礎的系統架構上加入不同演算法、強化學習等方式進行控制，最終成功將模擬的成果於實體雙足機器人實踐。

一、前言

在現今社會中，高齡化對社會的影響越來越劇烈，缺工已成為產業難以解決的問題之一，再加上 COVID-19 疫情的爆發，導致全世界的缺工問題更加嚴重。在這樣的環境下，如果能將機器人應用於一些工作場域或是代替人類執行某些任務，就能夠大大解決此問題。實際上目前許多工作場域都有慢慢引入機器人來替代簡易的工作，但可以發現這些機器人的工作場域常常侷限於平面地面，無法跨越樓層所帶來的限制，而雙足機器人則不受該種地形帶來的限制影響。

足型機器人在斜坡、階梯上往往較其它機器人有優勢。在空間沒有這麼寬廣的超市倉儲內，雙足機器人因其下半身的獨特性，能更好地在該環境下移動。近年來，越來越多公司投入雙足機器人的研發，如 Agility Robotic 的 Digit [1]、Boston Dynamics 的 Atlas [2]還有今年 Tesla 全新發表的 Optimus Gen 2 [3]，顯示出世界知名公司也都致力於發展雙足機器人領域。然而，國內對於此方面的投入卻相對十分稀少。最主要的原因在於此種機器人研究往往需要投入大量的資源才能夠實現。因此，本計畫之一大目標便是利用這有限的資源，進行雙足機器人的開發和研究，使其具備人機協作，甚至是不同機器人間的相互合作。

二、雙足機器人

1. 機構設計理念與目的

本團隊目前已開發雙足機器人的下半身機構如圖 1 與圖 2，同時也建立了雙足運動數學模型。為了提高行走效率、儲存能量以及釋放動能，參考了人類腿部結構與足弓進行設計。透過彈簧、馬達、連桿等各種元件的結合去建立了一個強健的機構，同時具備緩衝功能，以極大程度地減少機體損傷的風險。此外，步態模型的建立與感測器的回傳資訊實現了對機器人的精確掌控，使其能夠完成多種預期中的行為和動作。

為了符合提出的應用場域，雙足機構的機構設計標準是輕量化以及減少能

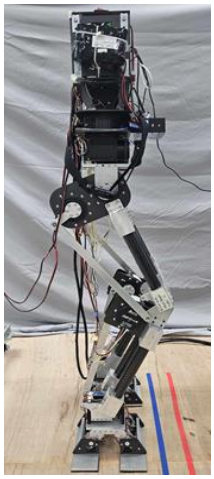


圖 1、機器人側視圖.

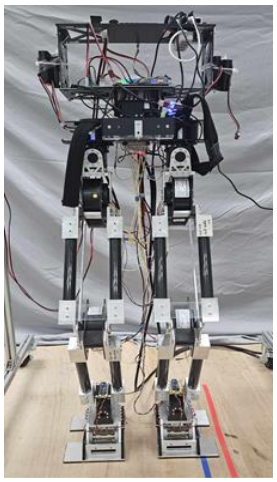


圖 2、機器人正視圖.

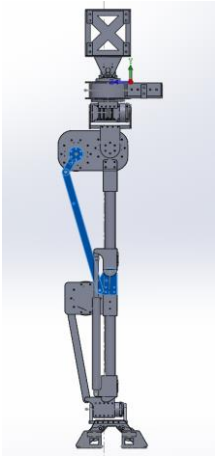


圖 3、機器人 CAD 圖.

量浪費，除此之外還需要有搬運重物的能力。透過模仿人類腿部的構造，讓雙足機器人在運動過程中，可降低機器人與環境互動時所造成的振動，並運用連桿機構傳動方式，使馬達能配置於臀部，提高雙足機器人腿部的重心位置，其目的是增加腿部的靈活性，且由於執行需要抬腿的運動時，提高腿部重心能降低旋轉慣性，同時也能降低馬達輸出時所消耗的能量，以增加馬達的壽命。

傳動方式上採用了四連桿機構如圖 3 反藍部分，透過連桿轉動使關節隨連桿距離變化產生轉動，如同人類透過肌肉收縮使關節彎曲。使用四連桿的優點有以下幾點，首先，可以將腿部的重心拉高，不需要將馬達配置在轉動關節位置上，有效降低雙足的旋轉慣性，馬達也相對省力，再者使用四連桿可以限制關節的轉動角度範圍，即使馬達突然失控也不會輕易造成機構受損，最後，雙足機器人上的四連桿皆為平行四連桿，也就是說在關節限制範圍內，控制關節的角度不需要考慮換算問題，馬達變動的角度即是關節屈伸的角度。

板材的部分主要以鋁件、POM、碳纖維板三者進行組裝，選材部分掌握一些原則，如轉動摩擦的關節或需攻牙處需用鋁件，其餘則以碳纖維或 POM 材質，連結關節轉動的部分使用光軸加上 C 型扣環，因其為受力點也是轉動點，因此轉軸需要是金屬件降低摩擦並確保不容易形變，而 C 型扣環是防止關節之間的相對滑動。板材雖然相對輕巧，但在強度上容易產生彎曲(bending)，由於機器人總長 126.4 公分，腿長即佔 102.1 公分，若只用板材勢必要選剛度高的材料，但這會相對增加機器人總重，因此採用管材並設計套筒去連接關節。為了達到輕量化的目的，這裡選用了碳纖維管，碳纖維管不只具備了輕巧優勢，在強度上也符合要求。

腳底板的設計如圖 4，為了降低與環境互動的震動和使雙足機器人行走能夠更像人類的動作，外觀的設計上模仿了人類的足弓，並在前腳尖與後腳跟的位置加上被動可旋轉的關節，原因是人在行走過程中前腳是腳跟先著地，而後腳起步時腳尖是最後離地，透過這個被動旋轉的機構，使行走上能更貼近人類步態。為達到被動旋轉的效果，採用了扭力彈簧如圖 5，扭力彈簧的彈力係數不可以過小，否則容易造成機器人的不平衡，又因市面上扭力彈簧規格固定，

表 1、雙足機器人規格

高度(cm)	126.4(腿長 102.1)
重量(kg)	21.9
自由度	13
材質	碳纖維、鋁、POM

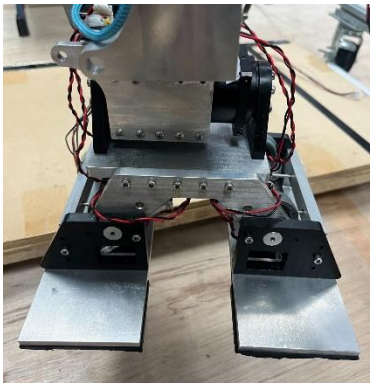


圖 4、腳板側視圖.

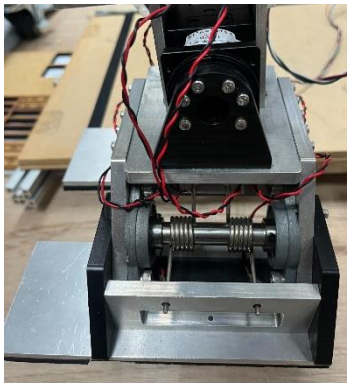


圖 5、扭力彈簧位置.

我們經過試驗方法去選擇合適的彈力係數，並在安裝時給彈簧一個預負載，使其更有減振效果。旋轉軸的部分使用軸承加光軸的方式，並使用板件固定扭力彈簧防止左右滑動如圖 5。

2. 系統架構:

A、硬體:

硬體除了設計理念中提到的部分，驅動主要由馬達來執行，馬達的部分採用智能伺服馬達(圖 6)和盤式馬達(圖 7)作為驅動，前者具有高精度和高可靠性的特性，同時支持多種通訊協定和控制模式，方便與其他元件協同工作，主要用於雙足機器人的 Yaw、Roll 方向關節，即腰與臀部的關節，後者是一種無刷直流馬達，具有高效率和低噪音的特性，並且內置了編碼器和驅動器，方便控制與調節，適合用於雙足機器人的 Pitch 方向關節，即腿部的關節，帶動連桿進行傳動。

感測器的部分主要有慣性量測單元(Inertial Measurement Unit, IMU)(圖 8)以及壓力感測器(圖 9)，前者用的是 Movella Xsens 開發的產品，其專精於慣性感測

圖 6、智慧型伺服馬達.

圖 7、盤式馬達.

表 2、使用馬達規格表

	智慧型伺服馬達		盤式馬達
重量(g)	732	340	900
扭力(N.m)	25.3 (at 24 V)	5.1 (at 24 V)	25 (at 48 V)
電壓(V)	24	10.0~14.8	24、48
最大輸出功率(W)	132	20	110

圖 8、Xsens MTI-7-DK IMU [4].

圖 9、薄膜壓力傳感器 [5].

模組的設計生產與多傳感器融合演算法(Sensor Fusion)，我們將其配置於腰部，蒐集機器人運動時的的尤拉角(Euler angle)、直線加速度與角加速度等資料，後者選用薄膜壓力傳感器，置於腳底板四個角落，蒐集機器人運動時腳底受力狀況。

B、軟體:

系統的架構主要由四個線程(Thread)組成，分別有 Main Thread、Pressure Thread、Motor Thread、IMU Thread。主線程負責控制每個線程的啟動與停止，接收感測器資料以產生運動命令，並透過 CAN bus 將其傳輸到馬達。另外三個線程用於接收馬達角度、速度、IMU 與壓力感測器的資料。

(1)、感測器資料前處理與分析:

由於雙足機器人運用不同馬達進行運作，在接收馬達回傳值時，容易遇見回傳時間不一的問題，因此在資料前處理上會適時的延遲，使馬達回傳時間盡量達成一致，方便後續閉迴路控制做使用。IMU 的回傳資料則是經解碼後透過將四元素轉換為尤拉角去使用，偵測機器人轉向之偏移幅度，作為步態修正之依據。壓力感測器的量測需要根據機器人總重量去選擇合適的電阻值，資料前處理上會先

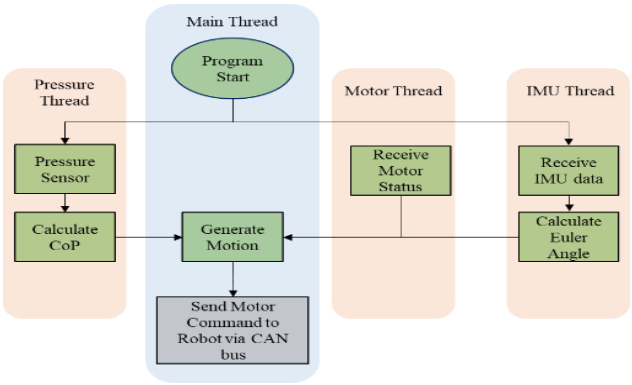


圖 10、雙足機器人之基礎系統架構

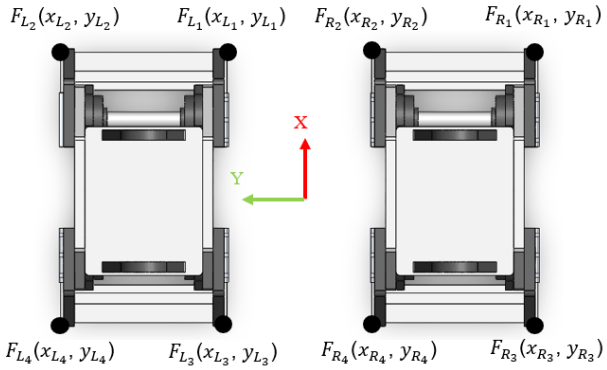


圖 11、壓力感測器安裝位置.

進行根據機器人實體情形進行雜訊濾波，防止因雜訊導致壓力中心位置的計算偏差擴大，處理過後的資訊會用來分析機器人模型的 Center of Pressure(CoP)偏移。壓力感測器裝置於機器人足部，左右腳的四個角落外圍各裝設一顆壓力感測器，用夾層方式於腳底板與踏墊間加裝壓力感測器，如此一來，在機器人腳部未接觸地面時，仍可繼續得到壓力偏移值，分別對雙腳各四個壓力感測器計算 CoP 之 x 與 y 位置，如(1)-(2)與圖 11 所示。

$$X_{CoP} = \frac{\sum_{i=1}^{4} F_{Ri} \cdot x_{Ri} + \sum_{i=1}^{4} F_{Li} \cdot x_{Li}}{\sum_{i=1}^{4} F_{Ri} + \sum_{i=1}^{4} F_{Li}}, \quad (1)$$

$$Y_{CoP} = \frac{\sum_{i=1}^{4} F_{Ri} \cdot y_{Ri} + \sum_{i=1}^{4} F_{Li} \cdot y_{Li}}{\sum_{i=1}^{4} F_{Ri} + \sum_{i=1}^{4} F_{Li}}。 \quad (2)$$

(2)、步態產生器數學模型:

為了讓雙足機器人能順利行走，一普遍且容易實現的步態模型為線性倒單擺(Linear Inverted Pendulum Model, LIPM)，只需使用有限的物理結構資訊，即可生成雙足機器人的步態。假設一個由點質量和無質量伸縮腿組成的倒單擺來模擬一條腿支撐身體時狀況，其質心位置 $x(t)$ 以及質心速度 $\dot{x}(t)$ 可以表示為:

$$x(t) = (x(0) - p_x)\, cosh\, cosh\left(\frac{t}{T_c}\right) + T_c\dot{x}(0) sinh\left(\frac{t}{T_c}\right) + p_x, \quad (3)$$

$$\dot{x}(t) = \frac{(x(0) - p_x)}{T_c}\, sinh\, sinh\left(\frac{t}{T_c}\right) + \dot{x}(0) cosh\left(\frac{t}{T_c}\right)。 \quad (4)$$

將此結論進一步推廣到三維的線性導單擺模型之中，分為 x-z 平面與 y-z 平面進行推倒，得出 x 與 y 方向上的質心位置與速度，以實現的步行控制。之後將 LIPM 得出的結果透過特定的轉換即可套用於我們的機器人身上，生成出我們所需要的動作。

為了能夠控制機器人的各關節，有必要進行一系列的數學推導來找出其中的角度與位置關係。在機器人領域中，順向運動學（Forward Kinematics）是一個重

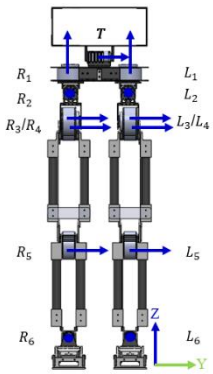


圖 12、機器人正視圖.

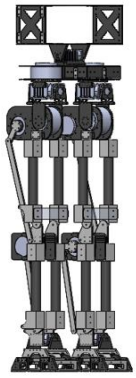


圖 13、機器人 45°視圖.

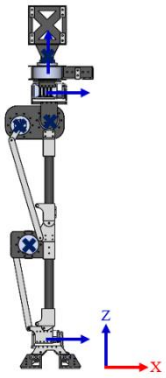


圖 14、機器人側視圖.

要的概念。它用來描述機械系統中各個部分相對於參考點或參考座標系的位置和方向，透過順向運動學，可從給定的機械系統關節角度中，計算末端點的位置和方向。為了建立出順向運動學，必須先定義出雙足機器人的 Denavit-Hartenberg（DH）表，DH 表的每一行需從機器人的前後關節找出資訊，其中包含四個參數：

1. θ_i： 繞著前一個 z 軸的角度，從舊的 x 軸到新的 x 軸。
2. d_i： 沿著前一個 z 軸到共同法線的偏移量。
3. a_i： 共同法線的長度
4. α_i： 繞著共同法線的角度，從舊的 z 軸到新的 z 軸。

透過 DH 表，我們可以在機器人的雙足上建立坐標系(圖 12-圖 14)。在設計的機器人身上，每隻腳各有 6 個自由度，分別是髖關節上的 Yaw(R_1, L_1)、Roll(R_2, L_2)與 Pitch 軸(R_3, L_3)、膝蓋的彎曲 Pitch 方向(R_4, L_4)以及腳踝的 Pitch(R_5, L_5)跟 Roll 方向(R_6, L_6)，其中 Z 軸為馬達旋轉方向，其中原點的部分設置在髖關節中 Yaw 方向(R_1, L_1)旋轉的馬達中心，表 3 呈現雙足機器人的 DH 參數。

表 3、雙足機器人的 DH 表.

i	θ_i	a_i	α_i	d_i
1	90°	L_1	90°	d_1
2	–90°	L_2	–90°	d_2
3	0	L_3	0	d_3
4	2.74°	L_4	0	0
5	-2.74°	L_5	90°	0
6	0	L_6	0	0

$$H_i^{i-1}=\begin{bmatrix}\cos\theta_i & -\cos\alpha_i\sin\theta_i & \sin\alpha_i\sin\theta_i & a_i\cos\theta_i\\ \sin\theta_i & \cos\alpha_i\cos\theta_i & -\sin\alpha_i\cos\theta_i & a_i\sin\theta_i\\ 0 & \sin\alpha_i & \cos\alpha_i & d_i\\ 0 & 0 & 0 & 1\end{bmatrix}, \tag{5}$$

$$H_{\mathrm{E}}^{6}=\begin{bmatrix}0 & 0 & -1 & 0\\ 0 & 1 & 0 & 0\\ 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}, \tag{6}$$

式(5)顯示的是轉換矩陣，其包含了每個關節的位置與方向資訊，透過將 DH 表中的參數帶入，可以得出機器人每個關節的轉換矩陣。式(6)表示踝關節 Roll 方向的馬達座標到腳底板座標的轉換矩陣，式(7)表示了原點的坐標轉換到末端點坐標:

$$H_{\mathrm{E}}^{0}=H_1^0H_2^1H_3^2H_4^3H_5^4H_6^5H_{\mathrm{E}}^6=\mathrm{P}_{\mathrm{E}}^0=\begin{bmatrix}r_{11} & r_{12} & r_{13} & P_{0E,x}\\ r_{21} & r_{22} & r_{23} & P_{0E,y}\\ r_{31} & r_{32} & r_{33} & P_{0E,z}\\ 0 & 0 & 0 & 1\end{bmatrix}。 \tag{7}$$

然而，在實際情況中，更多的時候我們已知機器人到達的目標位置，因此需要反過來求得關節角度，這時就需要運用到逆向運動學(Inverse Kinematics)，而每個關節角度可以透過式(7)左右等式相等來計算出，將線性導單擺模型規劃的步態軌跡反推回每個馬達所需執行的角度命令，實現各式各樣的動作。

三、不同任務之實現

1. 負重靜態與動態平衡即時控制

A. 簡介

本章節會介紹雙足機器人在運送貨品時所需的靜態與動態平衡技術。由於本團隊雙足機器人的行走步態是使用線性倒立擺模型控制機器人的行走軌跡，其為開迴路的控制方法，若直接將貨品放到雙足機器人上行走，機器人會因重心不穩而摔倒。因此本章節將針對此問題提出負重演算法以及平衡控制器，使雙足機器人再負重時得以穩定行走，使其順利完成運送貨品任務。

B. 控制系統架構

圖 15 為本章節的控制系統架構圖，首先雙足機器人是由姿態控制器來做出動態或是靜態的姿態，這裡所指的動態為 LIPM 產生的步態，靜態則是雙足機器人的初始穩定姿態。根據姿態控制器的輸出，平衡控制器會決定要採用單腳或者是雙腳的補償平衡，補償的多寡會由馬達當前狀態、慣性測量單元(IMU)和機器人腳底板裝設的壓力感測器來決定。若是在靜態的情形下，雙足機器人會根據壓

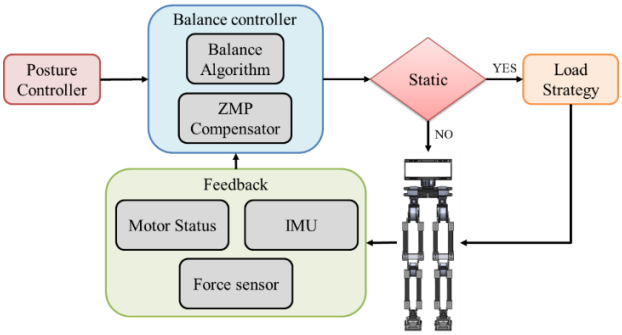


圖 15、負重平衡控制系統架構

力感測器來判斷雙足機器人是否被裝載貨品，根據情況啟用負載緩衝策略，使機器人不會因為裝載重物而失去平衡摔倒。最後總和所有控制器的輸出至雙足機器人每顆馬達做出理想的平衡姿態。

C. 平衡控制器

無論雙足機器人的狀態為動態或靜態，所有良好的動作都是基於雙足機器人在保持平衡的狀態下產生，因此保持平衡是不可或缺的要素。尤其對大型機器來說，因為重心較高，在動作的控制上也會變得更加困難，因此為了使雙足機器人有更好的表現，平衡控制器的存在成為關鍵。此章節所提出的平衡控制器分為兩個部分，第一部分為平衡演算法，第二部分為零力矩點(ZMP)補償器，而兩部分輸出的總和為平衡控制器的輸出。平衡演算法流程圖如圖 16，並分成靜態以及動態。

首先雙足機器人在靜態時通常保持著雙腳對齊貼平地面，且質心投影位置在雙腳中心的位置，以利接下來運動，然而外在環境的變化如地板傾斜或外力干擾，會使質心偏離雙腳中心不利於下一個動作，甚至最嚴重導致摔倒，為了解決這個問題，此演算法提供了適應環境變化的能力。根據兩隻腳的馬達角度回授，透過順向運動學求得兩隻腳的末端執行器(End effector)位置以及旋轉矩陣，在前述所提的靜止狀態下，左右腳的旋轉矩陣必定相同，由此可以得知雙腳末端執行器座標相對質心座標的旋轉情況，而 IMU 回授可得知質心座標傾斜多寡，為了使質心座標保持在平衡位置，此演算法將質心座標的傾斜角度乘上權重值後加入旋轉矩陣，末端執行器位置也會隨著傾斜而調整，再透過逆向運動學映射回各個馬達做即時補償，達到質心座標平衡的目標。

在動態的部分，步態週期分為單腳支撐和雙腳支撐兩種階段，支撐腳是行走保持平衡的關鍵，此演算法主要是補償單腳支撐階段，因為雙足機器人在行走時雙腳支撐是為了將重心轉移至另一支撐腳，其時間遠小於單腳支撐階段，因此演算法將轉移重心前半段的支撐腳視為後腳，而後半段的支撐腳視為前腳。由於 LIPM 產生的步態質心座標並不會轉動，因此可以根據 IMU 回授做補償。動態補償方式與靜態有兩點不同，第一是權重值較小，因為在行走情形下做較大幅度的

Algorithm 1 . Balance algorithm

Input:
the motors of left foot $\boldsymbol{\theta_i}$ for $i = L1,...,L6$,
the motors of right foot $\boldsymbol{\theta_j}$ for $j = R1,...,R6$,
the angles of IMU $\boldsymbol{\theta_k}$ for $k = r, p$
the angular velocity of IMU $\dot{\boldsymbol{\theta}}_{\boldsymbol{k}}$ for $k = r, p$
the weight of compensation $\boldsymbol{w}$
the thresholds of angular velocity $\boldsymbol{v_{up}}, \boldsymbol{v_{low}}$
Output:
$\boldsymbol{\theta_i}$ for $i = L1,...,L6$,
$\boldsymbol{\theta_j}$ for $j = R1,...,R6$

if static **then**
  $\boldsymbol{w} = 1$
  $\boldsymbol{R_l}, \boldsymbol{P_l}$ = Forward kinematics($\boldsymbol{\theta_i}$)
  $\boldsymbol{R_r}, \boldsymbol{P_r}$ = Forward kinematics($\boldsymbol{\theta_j}$)
  $\boldsymbol{R} = \boldsymbol{R_r} = \boldsymbol{R_l}$ because $\boldsymbol{R_l}$ equals to $\boldsymbol{R_r}$ in static
  $\boldsymbol{\theta_{base}} = Convert(\boldsymbol{R}) + \boldsymbol{w} \times \boldsymbol{\theta_k}$ *for base = roll, pitch*
  $\boldsymbol{P_l^{new}} = (x_l, y_l, z_l + L \times sin(\boldsymbol{\theta_{pitch}}))$
  $\boldsymbol{P_r^{new}} = (x_r, y_r, z_r - L \times sin(\boldsymbol{\theta_{pitch}}))$
  $\boldsymbol{R_{new}} = Convert(\boldsymbol{\theta_{base}})$
  $\boldsymbol{\theta_i}$ = Inverse kinematics($\boldsymbol{R_{new}}, \boldsymbol{P_l^{new}}$)
  $\boldsymbol{\theta_j}$ = Inverse kinematics($\boldsymbol{R_{new}}, \boldsymbol{P_r^{new}}$)
else
  if left foot support **then**
    $\boldsymbol{w} = 0.3$
    $\boldsymbol{R_l}, \boldsymbol{P_l}$ = Forward kinematics($\boldsymbol{\theta_i}$)
    $\boldsymbol{\theta_{base}} = Convert(\boldsymbol{R_l}) + \boldsymbol{w} \times \boldsymbol{\theta_k}$ *for base = roll, pitch*
    $\boldsymbol{R_l^{new}} = Convert(\boldsymbol{\theta_{base}})$
    $\boldsymbol{\theta_i}$ = Inverse kinematics($\boldsymbol{R_l^{new}}, \boldsymbol{P_l}$)
  else if right foot support **then**
    $\boldsymbol{w} = 0.3$
    $\boldsymbol{R_r}, \boldsymbol{P_r}$ = Forward kinematics($\boldsymbol{\theta_j}$)
    $\boldsymbol{\theta_{base}} = Convert(\boldsymbol{R_r}) + \boldsymbol{w} \times \boldsymbol{\theta_k}$ *for base = roll, pitch*
    $\boldsymbol{R_r^{new}} = Convert(\boldsymbol{\theta_{base}})$
    $\boldsymbol{\theta_j}$ = Inverse kinematics($\boldsymbol{R_r^{new}}, \boldsymbol{P_r}$)
  else if $|\dot{\boldsymbol{\theta}}_{\boldsymbol{k}}| > \boldsymbol{v_{up}}$ **then**
    Pause motion upon completing a period
    $\boldsymbol{w} = 0.6$
    Execute the static condition until $|\dot{\boldsymbol{\theta}}_{\boldsymbol{k}}| < \boldsymbol{v_{low}}$
  end if
end if

圖 16、平衡演算法

補償會導致質心速度方向改變太大，導致機器人摔倒，因此找到適合的權重值更有助於行走時的穩定性，根據實驗結果在行走時權重值為 0.3 最穩定。然而在一連串的動作下，雙足機器人可能會因外力導致質心速度改變偏離理想運動軌跡，因此若發生質心速度高於平常穩定行走的閥值時，演算法會在完成當前行走週期後將動作暫停，並進入暫時的靜態平衡直到質心穩定，此時的權重值不可與完全靜態時相同，因為此時的質心較不穩定，若權重值太大會導致過度補償，根據實驗結果權重值為 0.6 時可最快穩住質心。

在執行平衡演算法時，必須確保腳底板是貼平地面，因此提出零力矩點補償器來解決此問題。根據腳底板的壓力感測器所回授的類比訊號，可以通過(8)至(11)式獲得兩隻腳的零力矩點位置：

$$Z_v = \frac{(F_1 + F_4)}{(F_2 + F_3 + \varepsilon)}, \tag{8}$$

$$Z_x = \frac{L_x \times Z_v}{(1 + Z_v)} - \frac{L_x}{2}, \tag{9}$$

$$Z_h = \frac{(F_1 + F_2)}{(F_3 + F_4 + \varepsilon)}, \tag{10}$$

$$Z_y = \frac{L_y \times Z_h}{(1 + Z_h)} - \frac{L_y}{2}, \tag{11}$$

其中Z_x、Z_y為零力矩點座標，L_x、L_y為腳板的長與寬，ε為極小數值，其目的是為了防止(8)、(10)式的分母為零，因為將座標零點設定在腳板中心如圖 17，因此需要減去一半的腳板長寬。零力矩點補償器透過調整腳踝的 Roll 和 Pitch

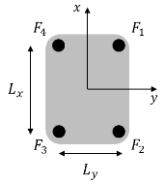


圖 17、零力矩點座標參數示意圖

方向的馬達來達成目標，控制方式是採用 PD 控制器如下(12)、(13)式：

$$\theta_i = K_i\left(Z_{i,y} - Z_{i,y}^d\right) + D_i\left(\dot{Z}_{i,y} - \dot{Z}_{i,y}^d\right), \tag{12}$$

$$\theta_j = K_j\left(Z_{i,x} - Z_{i,x}^d\right) + D_j\left(\dot{Z}_{i,x} - \dot{Z}_{i,x}^d\right), \tag{13}$$

其中$i = R6,\ L6$、$j = R5,\ L5$表式左右腳，因此補償器總共會有 4 組 PD 參數，針對腳踝四顆馬達進行角度調節。然而要同時找到 4 組最佳的參數並不容易，為了解決此最佳化問題，此章節使用元啟發式演算法根據設定好的訓練環境計算適性函數(Fitness function)找尋最佳解，而訓練環境和適性函數的設計將影響是否能找到最佳解的關鍵。

為了有效尋找，訓練環境由多階段的任務組成，其中包含地面旋轉、外力干擾以及負重行走，任務總長為 2500 個時間步長(Time step)，每個時間步長為 0.02 秒，三個任務長度分別 1000、500、1000 時間步長，為了縮短總訓練時間，當質心高度小於設定閥值時則判定雙足機器人摔倒提早結束訓練。在地面轉動的任務會分為三個階段，其分別為對 Roll 方向旋轉、對 Pitch 方向旋轉、對 Roll 和 Pitch 方向同時旋轉，旋轉方式如(14)式：

$$\theta(t)_{ground} = \frac{\pi}{18} \times \sin\left(\frac{t \times 2\pi}{T_g}\right), \tag{14}$$

其中T_g為地面轉動週期，長度為 250 個時間步長，在這時段內地面會依據設定方向旋轉正負 10 度。在外力干擾的任務中，設定在指定時間點對機器人施予 195 N 的外力，其方向分為 x 和 y 方向。負重行走任務是在指定時間點會負載重物，平衡後行走至時間結束，負載重量為 2 kg。

在適性函數的設計上，去計算左右腳每個時間步長在 x 與 y 的零力矩點座標位置與速度與理想狀態差的絕對值，最後將所有值總和作為其訓練適性值(Fitness value)，然而因為有提早結束訓練的機制存在，在被判定提早結束時，必須將剩餘的時間不長皆給予懲罰分數，否則在訓練時演算法會為了得到最小的適性值而選擇讓機器人摔倒。適性函數如下：

$$L_{x,i} = abs\left(Z_{l,x} - Z_{l,x}^d\right) + abs\left(\dot{Z}_{l,x} - \dot{Z}_{l,x}^d\right), \tag{15}$$

$$L_{y,i} = abs\left(Z_{l,y} - Z_{l,y}^d\right) + abs\left(\dot{Z}_{l,y} - \dot{Z}_{l,y}^d\right), \tag{16}$$

$$R_{x,i} = abs\left(Z_{r,x} - Z_{r,x}^d\right) + abs\left(\dot{Z}_{r,x} - \dot{Z}_{r,x}^d\right), \tag{17}$$

$$R_{y,i} = abs\left(Z_{r,y} - Z_{r,y}^d\right) + abs\left(\dot{Z}_{r,y} - \dot{Z}_{r,y}^d\right), \tag{18}$$

$$F = p \times (T - k) + \sum_{i=0}^{k} L_{x,i} + L_{y,i} + R_{x,i} + R_{y,i}, \tag{19}$$

其中$L_{x,i}$、$L_{y,i}$、$R_{x,i}$、$R_{y,i}$為左右腳 x 與 y 方向適性值，在(19)式中F為最後適性值，p為懲罰分數，T為總時間步長，k為完成之時間步長。

在訓練用的元啟發式演算法採用較為經典的粒子群演算法(PSO) [6]執行訓練，其演算法流程圖如圖 18，此算法是源於對生物群體行為的模擬，如鳥群或魚群的集體行為。它通過粒子在搜索空間中的移動，以找到最佳解。PSO 的原

Algorithm 2 Particle Swarm Optimization (PSO)

Initialize N particles' positions $\mathbf{X}_i$ and velocities $\mathbf{V}_i$ randomly
Initialize personal best positions $\mathbf{pbest}_i$ and global best position $\mathbf{gbest}$
while stopping criterion not met **do**
  for $i = 1$ to N **do**
    Evaluate fitness of particle $\mathbf{X}_i$: $f(\mathbf{X}_i)$
    if $f(\mathbf{X}_i) < f(\mathbf{pbest}_i)$ **then**
      Update $\mathbf{pbest}_i = \mathbf{X}_i$
    end if
    if $f(\mathbf{X}_i) < f(\mathbf{gbest})$ **then**
      Update $\mathbf{gbest} = \mathbf{X}_i$
    end if
    Update velocity of particle $\mathbf{V}_i$:

$$\mathbf{V}_i = \omega\mathbf{V}_i + c_1 r_1(\mathbf{pbest}_i - \mathbf{X}_i) + c_2 r_2(\mathbf{gbest} - \mathbf{X}_i)$$

    Update position of particle $\mathbf{X}_i$:

$$\mathbf{X}_i = \mathbf{X}_i + \mathbf{V}_i$$

  end for
end while

圖 18、PSO 演算法流程

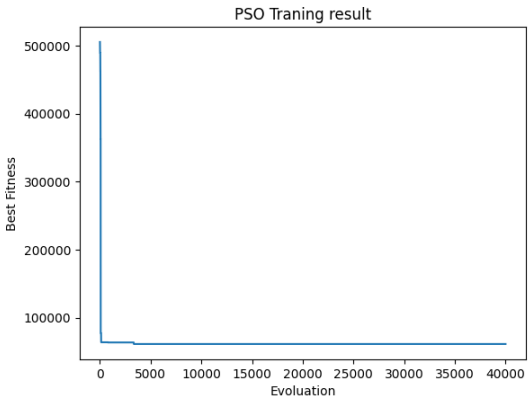


圖 19、PSO 10 次訓練平均適性值變化

理相對簡單，但卻能在許多問題中展現出強大的搜索能力。因為每次訓練的收斂結果會有所不同，也有可能陷入局部最佳解，為拿到最佳的參數，將透過數次的訓練並採用最佳適性值的參數作為零力矩點補償器的控制參數。訓練 10 次平均收斂結果如圖 19，在計算適性值的位置和速度單位使用毫米和毫米每秒，可根據訓練結果最佳適性值為 57138，在除以總時間步長後，平均每個時間步長所得之適性值為 22.85，由此可知偏移量很小，可以確保腳底板在執行平衡演算法時平貼於地面。

D. 負重策略

負重策略是模仿人在搬取重物時，會先藉由膝關節緩衝並藉由髖關節調整身體重心，以此快速達到穩定，為了幫助雙足機器人從未負重至負重的階段起到緩衝效果，並加速讓機器人取得平衡，策略流程圖如圖 20，此策略為快慢補償同時進行的策略，Knee strategy 為快速補償，Hip strategy 為慢速補償。

首先介紹 Knee strategy，此策略是根據壓力感測器是否有感受到外來重量而啟動，在偵測到外來重量時，Knee strategy 會透過快速改變膝關節角度幫助機器人做緩衝，降低外物衝擊對機器人造成的不穩定性，補償方法如(20)、(21)式：

$$\theta_{knee} = \beta \times a \times sin\left(\frac{s \times \pi}{2T_{knee}}\right) + B\,, \tag{20}$$

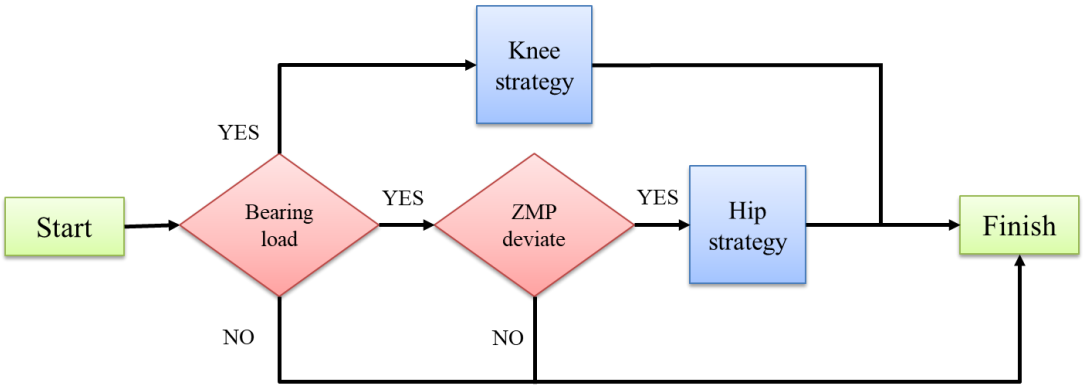


圖 20、負重策略流程圖

$$s_{t+1},\ a_{t+1},\ B_{t+1} = \begin{cases} s_{t+1} = 5,\ a_{t+1} = \theta_{t+1}^{pitch},\ B_{t+1} = B_t, & if\ \ |\theta_{t+1}^{pitch}| > |a_t| \\ s_{t+1} = s_t + 1,\ a_{t+1} = a_t,\ B_{t+1} = B_t, & else\ if\ \ s_t < T_{knee} \\ s_{t+1} = 1,\ a_{t+1} = \theta_{t+1}^{pitch},\ B_{t+1} = B_t + a_t \times \beta, & else\ \ s_t = T_{knee} \end{cases}, \quad (21)$$

其中θ_{knee}為補償θ_{R4}、θ_{L4}的角度，β為補償權重，a為當前傾角絕對值的最大值，s為迭代計數參數，T_{knee}為補償週期，B為累計過去補償值。這個策略分為三個階段，第一階段為穩定質心，此階段 IMU 回授質心座標傾角持續擴大，此時s維持定值而a持續更新，直到傾角減小，此時θ_{knee}會隨著a更新而快速增加，幫助快速穩定；第二階段為快速回正，此時通過迭代增加s而a維持定值，通過 sin 波的補償快速回到傾斜程度較低的狀態；第三階段為微幅調整，在s完成一次周期後重新計算，a會根據更新當前傾角值，B累計目前已補償角度，藉此回到最初未傾斜狀態。

Hip strategy 主要是透過調整髖關節的 pitch 方向馬達使雙足機器人重心保持在雙腳中心，補償方式如(22)式：

$$\theta_{accu}^{k+1} = \theta_{accu}^{k} + \gamma \times \frac{Z_{r,x} + Z_{l,x}}{2}, \quad (22)$$

其中θ_{accu}為 Hip 累計補償角度，γ為補償權重，此方法式根據零力矩點的位置做迭代累計補償，透過這個方式可讓重心穩定靠近雙足的中心點，雖然只能無限接近中心點無法到達，但只要讓重心足夠靠近中心點，雙足機器人就可以保持良好的平衡姿態。

兩種策略是以同時補償的方式使機器人快速回到穩定狀態，為了不讓兩種策略相互衝突，補償方法如(23)至(26)式：

$$\theta_{\mathrm{R3}}^{k+1} = \theta_{\mathrm{R3}}^{k} + \theta_{accu}^{k+1}, \quad (23)$$

$$\theta_{\mathrm{L3}}^{k+1} = \theta_{\mathrm{L3}}^{k} + \theta_{accu}^{k+1}, \quad (24)$$

$$\theta_{\mathrm{R4}}^{k+1} = \theta_{\mathrm{R4}}^{k} + \theta_{knee} - \theta_{accu}^{k+1}, \quad (25)$$

$$\theta_{\mathrm{L4}}^{k+1} = \theta_{\mathrm{L4}}^{k} + \theta_{knee} - \theta_{accu}^{k+1}, \quad (26)$$

在(25)、(26)式通過扣掉 Hip strategy 的補償角度，使雙足機器人在姿態上能保持平衡。

為了驗證本章節所提出的控制系統，本團隊在先在模擬環境運行測試，其流程如圖 21。在環境中，雙足機器人被賦予載運 2kg 的紅球前行的任務，可以看到機器人在保持平衡狀態後，順利運用平衡控制器與負重策略，在負重平衡後穩定的行走。

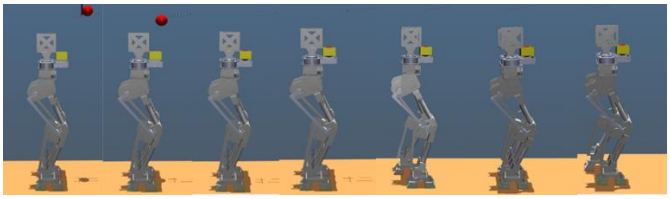


圖 21、負重平衡到行走流程

2. 上下樓梯任務設計與實現

A. 簡介

由於調整 LIPM 參數是一件非常繁瑣複雜且耗時的事情，且同樣的參數並不適用於各種情況，因此本計畫設計一款基於 Twin Delayed DDPG (TD3) [7, 8] 深度強化學習的回授控制系統，如圖 22，能夠讓機器人針對當前的外界情況預測出最適當的 LIPM 參數，並藉此達到穩定行走。

B. 基於人體前庭脊隨反射機制的可變質心高度 LIPM 步態模型

為了讓雙足機器人能夠更好得適應地形，這部份我們參考了人類在行走時的前庭反射機制 [9]，這項機制使人類能夠維持平衡並克服外在的各種干擾，因此我們將此套用在雙足機器人上，可以使機器人擁有自適應的穩定能力，從而避免摔倒的情況發生。在上下樓梯的任務當中，機器人的質心軌跡變化大致如圖 23 所示，在上樓梯行走時，系統反饋迴路被設計成驅動身體質心沿著樓梯向前移動，同時減少步幅長度以維持動態穩定性；相反地，在下樓梯行走時，身體質心將向後移動，步幅長度將增加以防止失去平衡，如此一來就能確保機器人在行走過程當中的穩定性。

在本作品當中，上下樓梯的步態模型是基於線性倒單擺模型 [10, 11]生成的，但傳統的 LIPM 模型假設機器人的質心高度是恆定不變的，使得此步態模型只適合用於雙足機器人於平地行走的情形，考量到這點，我們決定加以改變行走時的質心軌跡，使其更適應於垂直移動的任務。原始的 LIPM 軌跡相

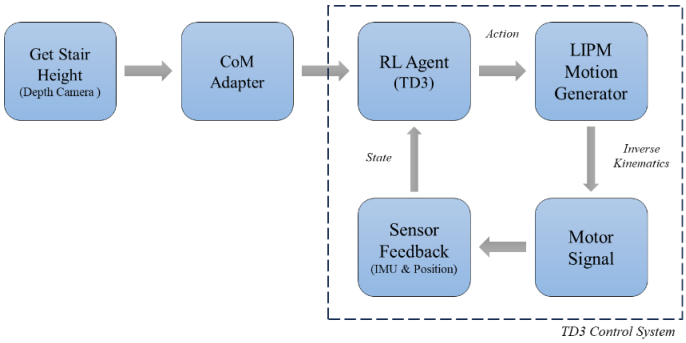
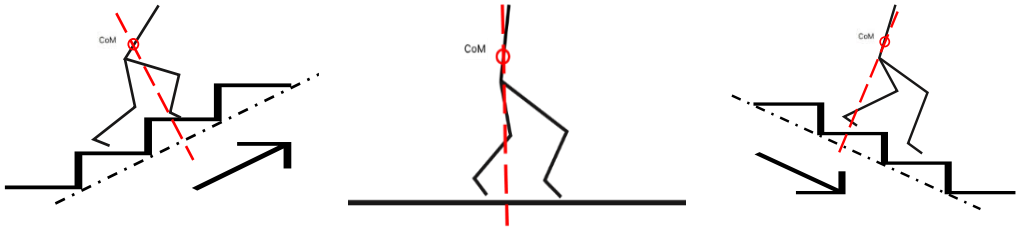


圖 22、TD3 回授控制系統



(a) (b) (c)

圖 23、機器人面對樓梯地形時的重心姿態。
(a)上樓梯 (b)行走於平面 (c)下樓梯

對於世界座標在垂直方向上的變化如式(27):

$$ {}^{w}foot_z(t) = \begin{cases} \frac{H}{2}(1-cos\theta), & for\ t_1 < t < t_2 \\ 0, & otherwise \end{cases}, \tag{27} $$

$$ {}^{R}foot_z = {}^{w}foot_z - {}^{w}CoM_z, \tag{28} $$

其中 H 是機器人的抬腳高度，t_1 及 t_2 分別是雙腳支撐及單腳支撐的時間，θ則為$2\pi(t-t_1)/(t_2-t_1), t \in (0,T)$，相對於機器人座標的表示即為式(28)。接下來為了適應行走時不同的高度，我們加入了一個變化項，這裡定義為式(29)，此變化項能夠使機器人抬腳的高度隨著行進中去做最佳的調整，最終可變質心高度的 LIPM 在垂直方向上的軌跡變化為式(31)所示，其雙腳軌跡圖則如圖 24。

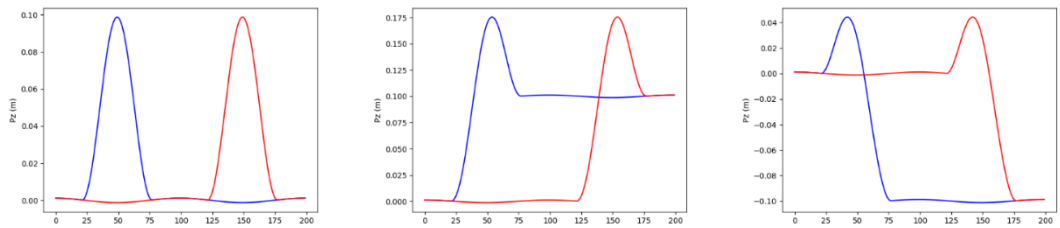
$$ {}^{a}CoM_z = -ACoM_z \cdot S(t), \tag{29} $$

$$ S(t) = \begin{cases} 0, & if\ t < t_1 \\ \frac{t - \sin(t)}{2\pi}, & otherwise \\ 1, & if\ t > t_2 \end{cases}, \tag{30} $$

$$ foot_z = {}^{w}foot_z - {}^{w}CoM_z - {}^{a}CoM_z。 \tag{31} $$

C. TD3 演算法

在強化學習當中有多種演算法可以使用，像是 DQN [12]、PPO [13]及 DDPG [14]等，這些演算法雖然都有各自的特點，但也有些不足的地方，像是 DQN 通常需要大量的樣本進行訓練導致學習過程較慢；PPO 容易陷入局部最佳解的情形；還有 DDPG 對超參數較為敏感不易調整，此外 DDPG 還有一個常常失敗的原因，就是訓練過程中所學習到的 Q 函數容易高估 Q 值，這導致了策略被破壞。TD3 則是為了解決此問題所開發的延伸演算法，其引入了三個方法來解決此一問題，第一點是 TD3 會學習兩個 Q 函數，並使用其中較小的 Q 值作為貝爾曼誤差函數當中的目標；第二是延遲策略的更新，TD3 對於策略的更新頻率低於 Q 函數的更新頻率，在 [7]當中建議更新 0 率為 1 比 2；第三則是平滑化目標策略，TD3 會在目標動作當中添加雜訊，並透過平滑 Q 值，使策略更難利用 Q 函數中的錯誤，透過這三個方法 TD3 彌補了 DDPG 中的不足。



(a) (b) (c)

圖 24、傳統 LIPM 與自適應 CoM LIPM 的雙腳高度在垂直方向上的變化。
(a)傳統 LIPM (b)改良式 LIPM 上樓梯 (c)改良式 LIPM 下樓梯

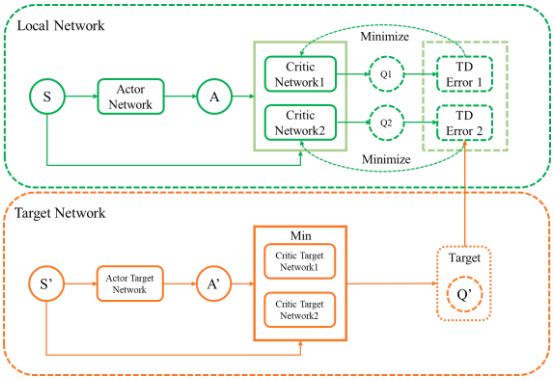


圖 25、TD3 架構圖

接下來看一些 TD3 當中重要的方程式，首先是目標策略平滑的部分，組成 Q-learning target 當中的動作都是基於目標策略$\mu_{\theta_{targ}}$，TD3 在動作的每個維度上都會添加 clipped noise，使得目標動作會被限定在有效的動作範圍之內，因此目標動作被寫作:

$$a'(s') = clip\left(\mu_{\theta_{targ}}(s') + clip(\epsilon, -c, c), a_{Low}, a_{High}\right), \epsilon \sim N(0, \sigma)， \quad (32)$$

平滑目標策略的本質上是作為此演算法中的正則化器，它解決了 DDPG 當中的錯誤估計尖峰。

其次是 clipped double-Q learning，這兩個 Q 函數都使用單個目標，該目標是使用兩個 Q 函數當中較小的一個來作計算:

$$y(r, s', d) = r + \gamma(1 - d) \min_{i=1,2} Q_{\phi_i, targ}\left(s', a'(s')\right)， \quad (33)$$

然後兩者透過回歸到此目標來學習:

$$L(\phi_1, D) = \underset{(s,a,r,s',d)\sim D}{\mathrm{E}}\left[\left(Q_{\phi_1}(s,a) - y(r,s',d)\right)^2\right]， \quad (34)$$

$$L(\phi_2, D) = \underset{(s,a,r,s',d)\sim D}{\mathrm{E}}\left[\left(Q_{\phi_2}(s,a) - y(r,s',d)\right)^2\right]。 \quad (35)$$

使用較小的 Q 值做為目標值，然後逐步回到該值，有助於避免 Q 函數被高估。最後策略會以最大化Q_{ϕ_1}為目標來學習:

$$\max_{\theta} \underset{s\sim D}{\mathrm{E}}\left[Q_{\phi_1}\left(s, \mu_\theta(s)\right)\right]。 \quad (36)$$

Algorithm 3 Twin Delayed DDPG

Input: initial policy parameters θ, Q-function parameters ϕ_1, ϕ_2, empty replay buffer $\mathcal{D}$
Set target parameters equal to main parameters $\theta_{\text{targ}} \leftarrow \theta$, $\phi_{\text{targ},1} \leftarrow \phi_1$, $\phi_{\text{targ},2} \leftarrow \phi_2$
repeat
  Observe state s and select action $a = \text{clip}(\mu_\theta(s) + \epsilon, a_{Low}, a_{High})$, where $\epsilon \sim \mathcal{N}$
  Execute a in the environment
  Observe next state s', reward r, and done signal d to indicate whether s' is terminal
  Store (s, a, r, s', d) in replay buffer $\mathcal{D}$
  If s' is terminal, reset environment state.
  if it's time to update **then**
    for j in range(however many updates) **do**
      Randomly sample a batch of transitions, $B = \{(s, a, r, s', d)\}$ from $\mathcal{D}$
      Compute target actions

$$a'(s') = \text{clip}\left(\mu_{\theta_{\text{targ}}}(s') + \text{clip}(\epsilon, -c, c), a_{Low}, a_{High}\right), \quad \epsilon \sim \mathcal{N}(0, \sigma)$$

      Compute targets

$$y(r, s', d) = r + \gamma(1 - d) \min_{i=1,2} Q_{\phi_{\text{targ},i}}(s', a'(s'))$$

      Update Q-functions by one step of gradient descent using

$$\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s',d)\in B} \left(Q_{\phi_i}(s,a) - y(r,s',d)\right)^2 \quad \text{for } i = 1, 2$$

      if j mod `policy_delay` $= 0$ **then**
        Update policy by one step of gradient ascent using

$$\nabla_\theta \frac{1}{|B|} \sum_{s\in B} Q_{\phi_1}(s, \mu_\theta(s))$$

        Update target networks with

$$\phi_{\text{targ},i} \leftarrow \rho\phi_{\text{targ},i} + (1 - \rho)\phi_i \quad \text{for } i = 1, 2$$
$$\theta_{\text{targ}} \leftarrow \rho\theta_{\text{targ}} + (1 - \rho)\theta$$

      end if
    end for
  end if
until convergence

圖 26、TD3 Pseudocode [8]

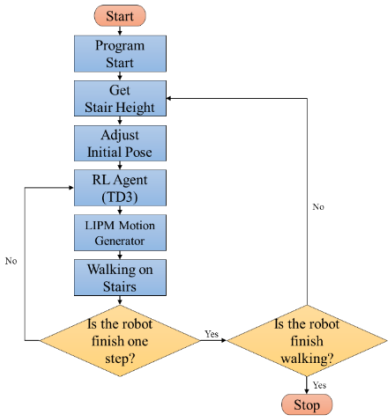


圖 27、系統流程圖

D. TD3 控制系統

為了要增強機器人在爬樓梯過程當中的穩定性及強健性，本作品將前段所提到的 TD3 強化學習構建成一個回授系統並用於此任務，整個系統流程如圖 27。在策略開始運作後，機器人首先會先獲取前方地形的高度，或是由操作者手動輸入，機器人會根據此數值來調整自身的 CoM 高度，藉以達到最穩定的姿態，接著再開始行走，行走的動作指令如前述是由 LIPM 模型生成的，再透過逆向運動學映射到每顆馬達上，並且在行走的過程當中蒐集外在的環境資訊，這些資訊會作為當前 RL agent 的狀態，接著 RL agent 會預測出下個時間點的動作並丟進 LIPM 當中，如此反覆以此達到即時的回授控制。

E. RL 強化學習架構

RL 作為此任務當中的重要角色，這一章節將說明 RL 的狀態以及獎勵函數等定義。

圖 28 為 RL 的流程圖，其中模型的輸入及輸出如下說明，環境狀態，也就是模型的輸入如式(37)，

$$\begin{gathered} s_t = [Q_{base},\ a_{base},\ p_{motor}, H_{stair}]\ , \\ Q_{base} = [w,i,j,k]^T\ , \\ a_{base} = [a_x, a_y, a_z]^T\ 。 \end{gathered} \tag{37}$$

其中的Q_{base}是在機器人 base 所量測到的四元數，a_{base}是 base 在 xyz 方向上的加速度，p_{motor}是每顆馬達當下的角度位置，H_{stair}則是前方樓梯的高度，訓練過程中有將H_{stair}做強化，以便在 RL 訓練當中更有效地辨認出此資訊，因為樓梯高度會對行走的過程產生非常大的影響。

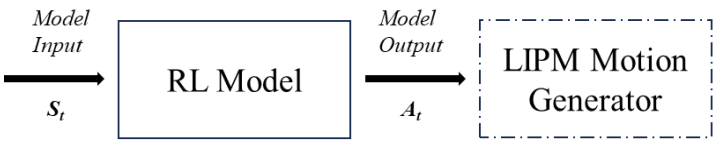


圖 28、RL 流程圖

另外模型的輸出則如式(38):

$$a_t = [B_R, B_L, S_R, h_R, h_L, H_R, H_L, A_R, A_L]。 \quad (38)$$

先前有提到 RL 的輸出動作會作為 LIPM 的輸入參數，因此這些數值都與機器人的行走相關，其中的 B 代表步行過程中機器人左右擺動的幅度、S 是跨步的長度、h 是抬腳過程中髖關節擺動幅度、H 是抬腳的高度、A 則是腳踝在 pitch 方向上傾斜的角度。

RL 的獎勵函數更是重要的一環，定義好的獎勵函數能夠讓訓練過程更加有效率，在此任務中須考量到的層面有很多，主要都跟機器人的狀態有關，首要目標還是希望機器人在執行任務過程當中能夠穩定且高效，因此本作品定義的獎勵函數如下:

$$R = w_1 R_{CoP} + w_2 R_{move} + w_3 R_{stable} + w_4 R_{healthy}， \quad (39)$$

$$R_{CoP} = (1-P)\cdot\left(-\sqrt{x_{l_{CoP}}^2 + y_{l_{CoP}}^2}\right) + P\cdot\left(-\sqrt{x_{r_{CoP}}^2 + y_{r_{CoP}}^2}\right)， \quad (40)$$

$$w_2 = [w_{21}, w_{22}, w_{23}]， \quad (41)$$

$$R_{move} = w_{21} x_{distance} - w_{22}|y_{deviation}| - w_{23}|\theta_{yaw}|，$$

$$R_{stable} = |\alpha_x| + |\alpha_y| + |\alpha_z|， \quad (42)$$

$$R_{healthy} = \begin{cases} 1, & for\ every\ timestep \\ -10, & if\ robot\ fall\ down \\ 20, & if\ mission\ success \end{cases}。 \quad (43)$$

本實驗設計的獎勵函數分成四個部分，第一是R_{CoP}，這項獎勵是用來評估機器人的重心位置是否落在腳底板的穩定區域內，因為若當機器人的重心位置離開了穩定多邊形區域，機器人極有可能會摔倒；第二是R_{move}，主要用來鼓勵機器人能夠向前行走，並且要避免發生 y 方向或 yaw 方向的不自然偏轉，其中的w_2是分別的權重，會在下一章進行說明；第三是R_{stable}，這是一個懲罰項，用來衡量機器人 base 的加速度，對於穩定行走來說，我們當然希望加速度會越小越好，因此給定此一懲罰；第四是$R_{healthy}$，此獎勵項有兩個功用，其一是彌補其他懲罰項所帶來的負值，其二是在任務完成或失敗的狀態下給予獎勵或懲罰，在本任務當中失敗的定義就是機器人摔倒或是偏轉太多，成功走上階梯則定義為成功的狀態。

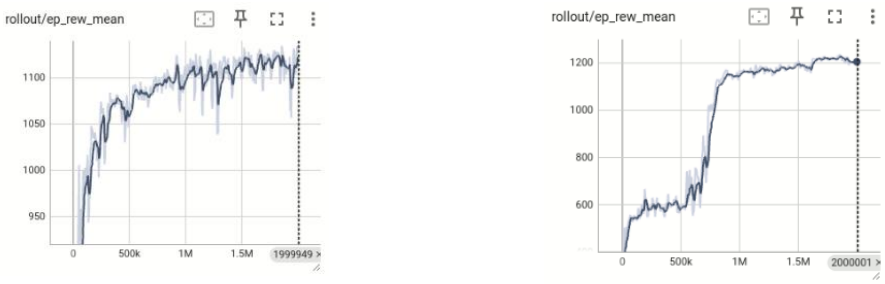
F. 實驗與模擬

(1)、訓練過程

上一章所提到的獎勵函數當中的參數如表 4，負號代表的即是懲罰項，這些參數的選擇是考量到各獎勵項之間數值大小的差異而給予不同的倍數，也有

表 4.獎勵函數中的參數

w_1	2	w_{23}	-5
w_{21}	0.5	w_3	-1/3
w_{22}	-2	w_4	1



(a) (b)

圖 29、訓練過程中的獎勵變化。(a)上樓梯 (b)下樓梯

根據訓練情形做適當調整，方才得到這組最佳參數。訓練過程中的獎勵變化如圖29，可以看到整體的獎勵都是上升的趨勢，且訓練到後期皆有收斂到一個穩定值。

(2)、模擬結果

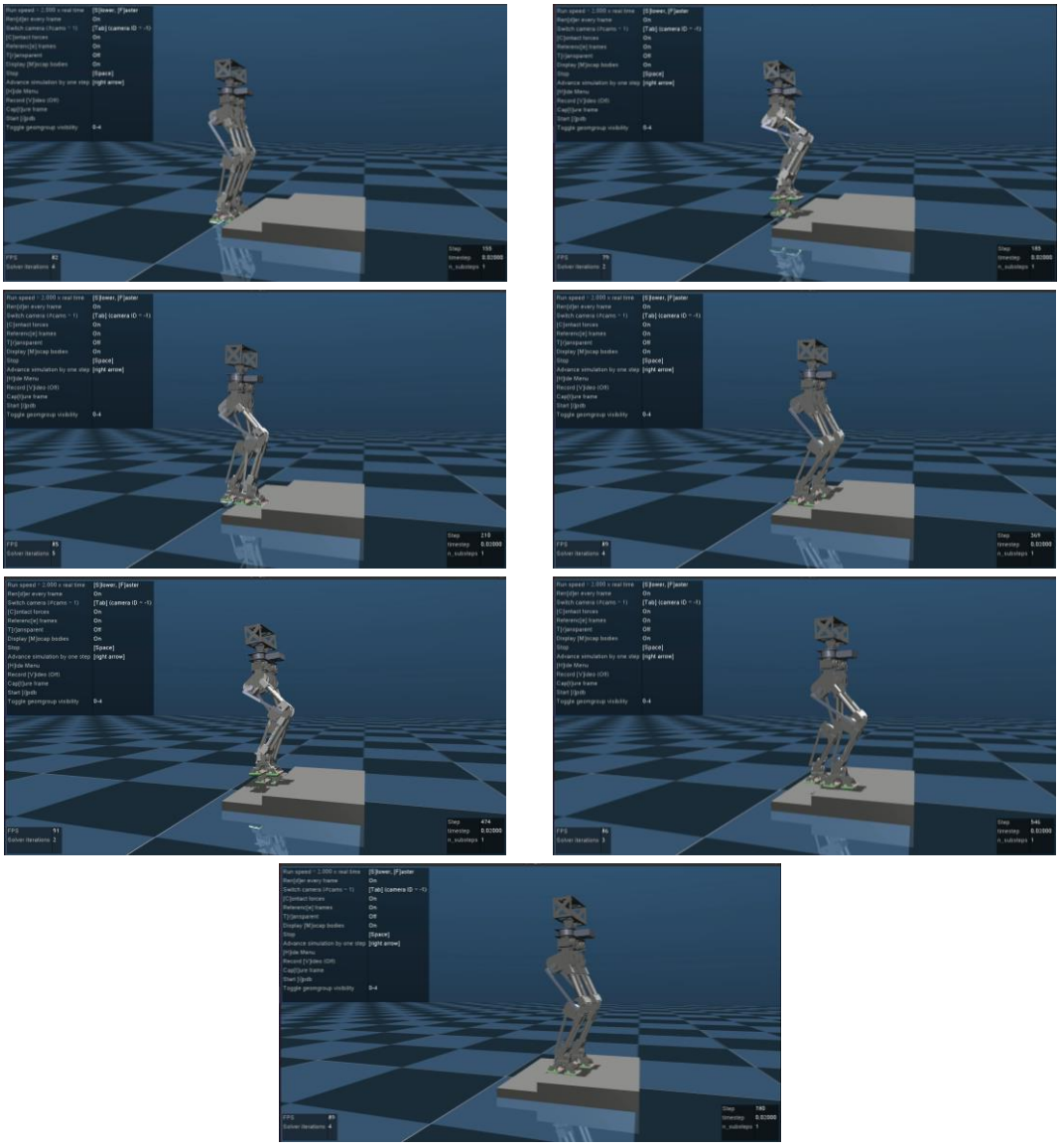


圖 30、在模擬器當中模擬上樓梯任務。

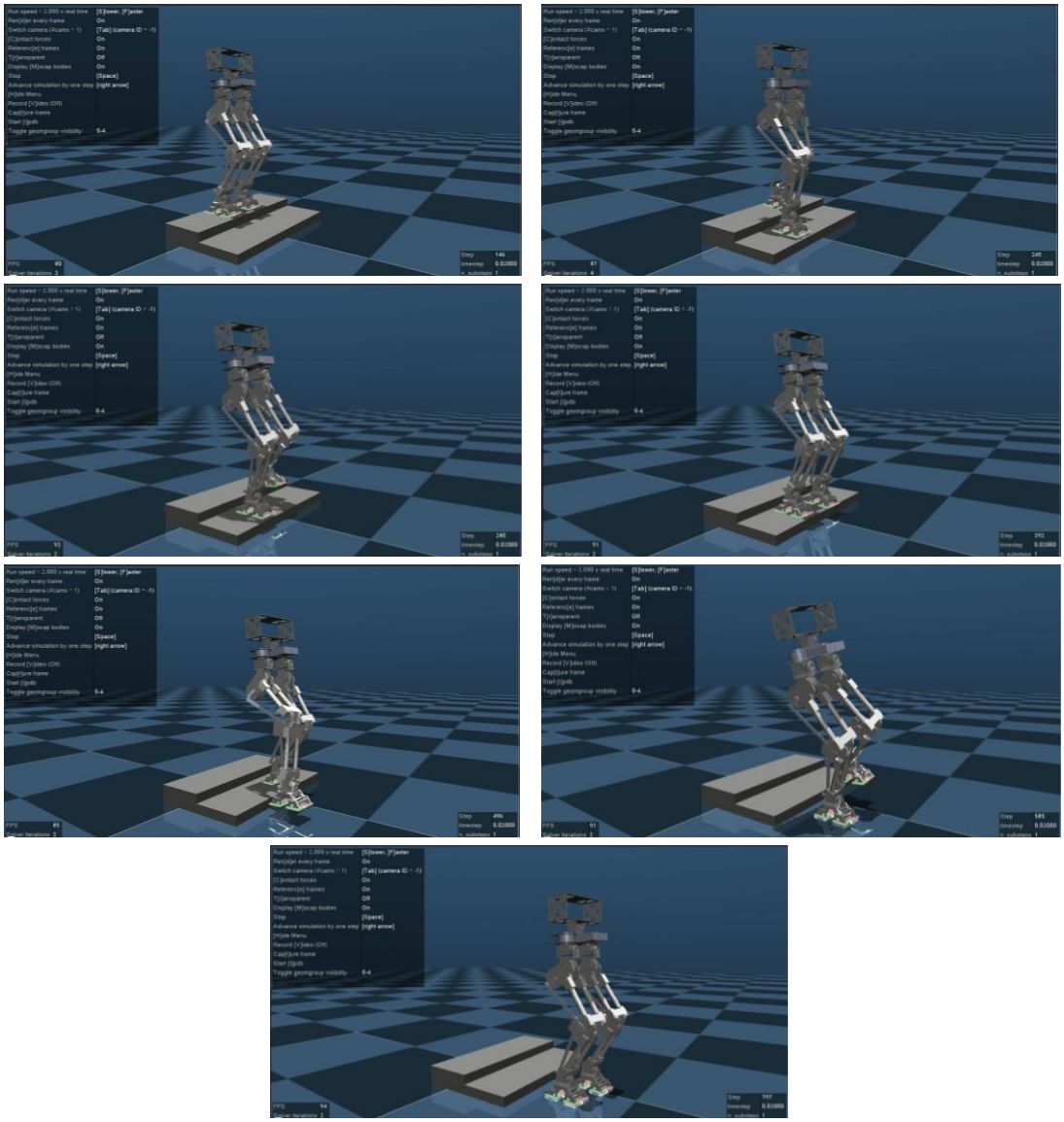


圖 31、在模擬器當中模擬下樓梯任務。

3. 行走於不平坦地形

A. 簡介

一個穩定的步態可以使機器人順利的在平面上行走。然而在實際的環境中，機器人將受到不同程度外界的影響，地面可能會有些微起伏，也可能有微小障礙物的存在，這些情況都會使機器人預期的行走產生偏差，進而導致跌倒。地面可能會有輕微起伏，也可能有小障礙物。這些情況都會導致機器人的預期行走出現偏差，進而導致摔倒。因此，在本任務的目標中，希望能讓機器人能夠抵銷上述提及的干擾所造成的擾動，使其在不平坦地面仍能像平坦地面行走。

當機器人在凹凸不平的地面或有障礙物的表面上行走時，機器人的腳會與地面接觸並產生一定的角度。如果傾斜角向上，機器人會按照原來的步態向後倒下；同樣，如果傾斜角向下，機器人會向前倒下。透過對各個關節進行軌跡的重新計

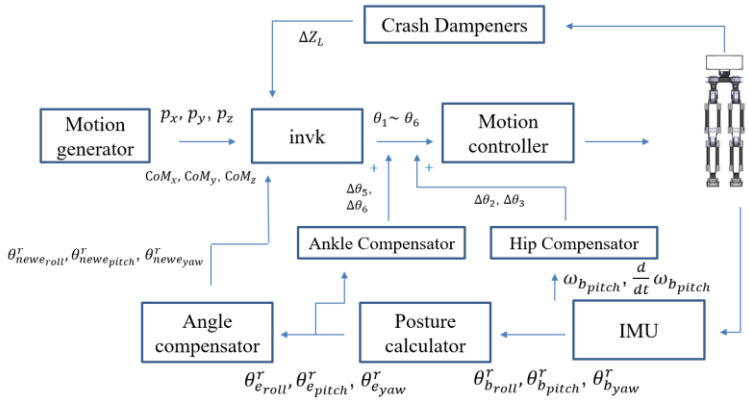


圖 32、不平坦地形對策架構圖

算以及補償關節的角度，使其上方的關節能讓 CoM（質量中心）保持在平面上行走的姿勢。

圖 32 為不平坦地形實現的簡易架構圖。機器人首先會基於事先產生好的 LIPM 動作檔產生一連串的馬達角度去完成行走動作，接著機器人會透過獲取放置在基座上 IMU 的角度姿態去進行末端點的角度計算以獲取地形的傾斜資訊，並透過當前機器人軀幹的傾斜角度去對應適應的傾斜角度去進行微調，之後將更新的資訊與走平地的步態去進行結合生成出新的適應當前地形的動作軌跡檔，以橫越複雜地形。同時，對於適應未知地形中可能會造成的擾動，如為了減少地面的衝擊而使用了碰撞緩衝器，以及髖關節與腳踝關節的角度補償，亦與主要方法一起結合，獲得有效的成果。

B. 地形偵測

實際的地形千變萬化，各種類型的場景無法一一舉例，但可以先以簡單的方式對其進行簡化分析，如圖 33 所示。從圖中可以看到，腳踩上的地面為兩個不同的地形障礙物，然而腳的擺放位置與角度卻十分相近，因此可以將這兩種地形視為同一種狀態。腳的旋轉方向可以分為 3 個自由度，分別為沿著縱軸旋轉的 roll 方向、沿著橫軸旋轉的 pitch 方向以及沿著樞軸旋轉的 yaw 方向。由於這三個自由度互相垂直的關係，可以視為互不干擾，所以可以對其進行分解分開計算再合併。

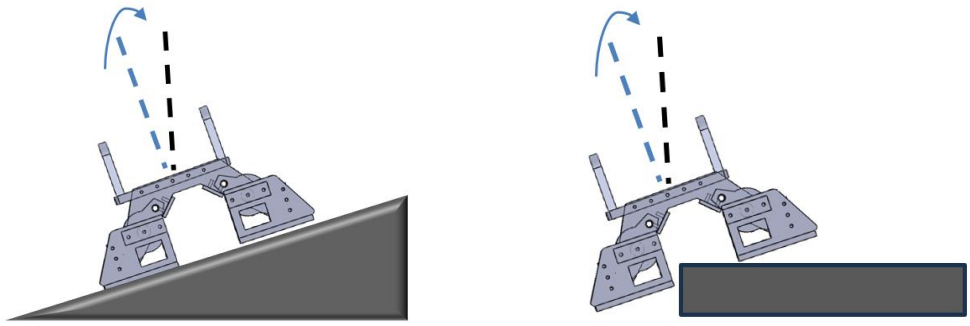


圖 33、地形示意圖

為了能對適應不同的地形，需要去偵測當前腳底板對於世界坐標系的姿態，而腳底板的姿態可以經由放置在基座上的 IMU 去推算。首先需要將 IMU 所測量到的 roll、pitch 和 yaw 方向角轉換成旋轉矩陣：

$$R_b = R_{b\_yaw}(\phi)R_{b\_pitch}(\theta)R_{b\_roll}(\varphi)$$
$$= \begin{bmatrix} \mathrm{C}\theta\,\mathrm{C}\phi & \mathrm{S}\varphi\,\mathrm{S}\theta\,\mathrm{C}\phi - \mathrm{C}\varphi\,\mathrm{S}\phi & \mathrm{C}\varphi\,\mathrm{S}\theta\,\mathrm{C}\phi + \mathrm{S}\varphi\,\mathrm{S}\phi \\ \mathrm{C}\theta\sin\phi & \mathrm{S}\varphi\,\mathrm{S}\theta\,\mathrm{S}\phi + \mathrm{C}\varphi\,\mathrm{C}\phi & \mathrm{C}\varphi\,\mathrm{S}\theta\,\mathrm{S}\phi - \mathrm{S}\varphi\,\mathrm{C}\phi \\ -\mathrm{S}\theta & \mathrm{S}\varphi\,\mathrm{C}\theta & \cos\varphi\,\mathrm{C}\theta \end{bmatrix}, \quad (44)$$

其中R右下標的 b 代表 base(基座)的旋轉矩陣，$\mathrm{C}\theta$代表$\cos\theta$、$\mathrm{S}\theta$代表$\sin\theta$，其他以此類推。機器人腿部以機器人 base 為坐標軸原點的機器人坐標系的末端座標和旋轉矩陣可透過機器人每個馬達關節的角度得出：

$$H_E^0 = H_1^0 H_2^1 H_3^2 H_4^3 H_5^4 H_6^5 H_E^6 = P_E^0 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & P_{0E,x} \\ r_{21} & r_{22} & r_{23} & P_{0E,y} \\ r_{31} & r_{32} & r_{33} & P_{0E,z} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (45)$$

$$\text{where } R_e = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix},$$

其中，R_e是末端點相對於基座的旋轉矩陣，將兩個旋轉矩陣相乘來獲得末端點在世界坐標系的旋轉矩陣，然後透過式(46)-(49)計算出末端點在世界坐標系中 roll、yaw、pitch 三個方向的角度，即可得出末端點的當前姿態。

$$R_{be} = R_e R_b \quad (46)$$

$$\theta_{e_{roll}} = tan^{-1}\left(\frac{r_{32}^{be}}{r_{33}^{be}}\right) \quad (47)$$

$$\theta_{e_{pitch}} = tan^{-1}\left(\frac{-r_{31}^{be}}{\sqrt{r_{32}^{be\,2} + r_{33}^{be\,2}}}\right) \quad (48)$$

$$\theta_{e_{yaw}} = tan^{-1}\left(\frac{r_{21}^{be}}{r_{11}^{be}}\right) \quad (49)$$

C. 穩定補償

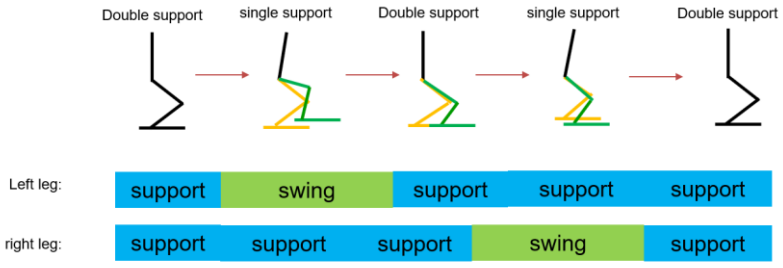


圖 34、行走週期

機器人行走二步的週期如圖 34 所示，可以看出機器人的左右腿分別會經過 support(支撐腳)以及 swing(擺動腳)的階段。當機器人所屬在未知的環境中，其對接下來的地形是未知的，因此其會採取行走在平地上的步態軌跡。從圖中可以看出，若是前方地面為起伏地形時，當機器人從 single support 轉換為 double support 時，擺動腳將會接觸到地面，並造成機器人的不穩定。為了檢測擺動腳何時碰觸到地面，可以使用安裝在腳底板上的壓力感測器去偵測。一旦壓力感測器偵測到數值，即可立即去計算末端點的姿態，以便之後能及時去重新規劃行走的軌跡，使其能符合當前接觸的地形。

在重新規劃軌跡之前，機器人即有可能就因為突如其來的接觸而造成跌倒，因此需要另外使用一連串的策略去進行平衡補償。設機器人的行走一步的週期為 T，雙腳支撐時間的比例為 kdsp(即圖 34 中擺動的那隻腳在一步中支撐地面時間的比例)，則即可推算出正常平地行走中，機器人擺動腳落地的期望時間$t_2 = T - \frac{k\,dsP}{2}$，透過偵測壓力感測器讀到數值的時間與落地期望時間的對比，可以判斷出下一步地面相對於前一步位置的水平高低以利於反應應對。此外，由於壓力感測器平均的分佈在腳底的四個端點，可以透過先讀取到哪一個值去推測判斷當前的地形，如腳尖分布的壓力感測器先讀值則極可能代表當前的地勢偏向上坡，此時即可對機器人的腳踝角度進行事先補償，盡可能地讓腳底能快速貼平地面，以便盡快掌握當前的地形資訊。

$$\Delta\theta_5 = k_1\left(\frac{1}{1+e^{-\theta_{e_{pitch}}}} - \frac{1}{2}\right)， \tag{50}$$

$$\Delta\theta_6 = k_2\left(\frac{1}{1+e^{-\theta_{e_{roll}}}} - \frac{1}{2}\right)。 \tag{51}$$

式(50)-(51)計算了需要去補償的腳踝角度，$\Delta\theta_5$和$\Delta\theta_6$分別為需要去快速適應的 pitch 和 roll 方向的補償變化量，k_1和k_2為增益的係數。

距離機器人的軀幹最近的關節為髖關節，其離末端點的距離相對於其他關節馬達的距離也最遠，造成的力矩也會相對較大，因此髖關節的角度變動對於機器人的重心轉移會有明顯的影響。當我們在讓腳踝快速適應地面時，可能會造成機器人的不穩定，此時就須要對髖關節的角度進行相對應的補償，其相關式子如式(52)-(53)所示:

$$\Delta\theta_3 = G_1\omega_{b_{pitch}} + G_2\frac{d}{dt}\omega_{b_{pitch}}， \tag{52}$$

$$\Delta\theta_2 = G_3\omega_{b_{roll}} + G_4\frac{d}{dt}\omega_{b_{roll}}。 \tag{53}$$

其中，ω_b為機器人基座上的角速度，$\frac{d}{dt}\omega_b$為基座上的角加速度，G_1~G_4為比例係數。突然的角速度或角加速度變化可能會導致失去平衡，透過對這些狀態去做偵測，可以預測機器人重心偏移方向，進而去做補救的措施。

由於機器人行走時我們給定的是各個關節的馬達應旋轉的角度，故在擺動腳接觸到地面時會產生很大的衝擊，此衝擊不但會造成機器人行走的不穩定，同時也會對馬達造成程度不一的磨損，進而加速減少其使用壽命。因此，需要對這部分進行緩衝處理使其著陸更加柔順。我們在接別人丟過來的球時，會將手向後彎曲以便吸收並緩衝衝擊，同理，我們可以將此想法應用到機器人身上，在腳與地面接觸時，透過動態的調整腳的高度，在高衝擊時將其略為彎曲以緩衝地面造成的反作用力。

$$\Delta Z_L = k_3 f_L \text{，} \tag{54}$$

$$\Delta Z_R = k_3 f_R \text{，} \tag{55}$$

其中ΔZ為彎曲所減少的腳長度，下標的L和R分別表示左腳和右腳，f為下標所屬的腳四個壓力感測器的合力值，而k_3為增益係數。接著將計算出的變化量ΔZ套入逆向運動學進行計算，即可獲得更新後的θ_3、θ_4、θ_5。

最後，對於獲取的末端點的姿態，將對其進行基於基座角度的微調。由於原始的步態是依據平地去產生的，因此儘管將其軌跡去適應地形的起伏，仍然會有一些誤差。透過追蹤實際與期望的基座角度變化，可以獲得最適合的姿態角度。

$$\theta_{newe_{pitch}} = \theta_{e_{pitch}} + k_4 \cdot \left(\theta^d_{b_{pitch}} - \theta^r_{b_{pitch}}\right) \text{，} \tag{56}$$

$$\theta_{newe_{roll}} = \theta_{e_{roll}} + k_5 \cdot \left(\theta^d_{b_{roll}} - \theta^r_{b_{roll}}\right) \text{，} \tag{57}$$

其中式(54)表示的是 pitch 方向的調整，式(55)則為 roll 方向的調整。θ右上標的d表示為期望的基座角度，θ右上標的r表示為實際的基座角度，k_4以及k_5為增益參數，可以透過智慧型演算法去尋找最佳解。

D. 模擬展示

對於機器人在不同的地形中進行行走在模擬中進行了一些測試，下面分成幾個不同的場景分別測驗各種情況下的結果。

(1)、不連續地面

圖 35 顯示了機器人走下高度約 1.2 公分的不連續地面。從圖 36 以及圖 37 中可以看出，一開始的馬達角度與走平地時無異，然而當腳在接觸到地面的時候，由於與期望的平地地形產生誤差，因此機器人能快速的針對地形的起伏去作出調整，馬達的軌跡角度也開始與平地的步態出現差異。最後機器人能成功地走下地面。

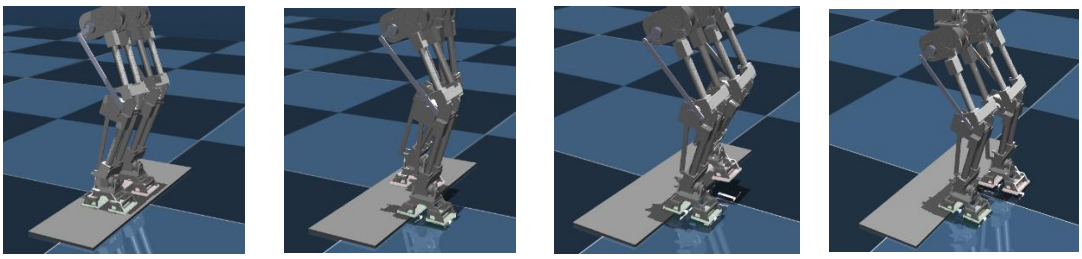


圖 35、不連續地面

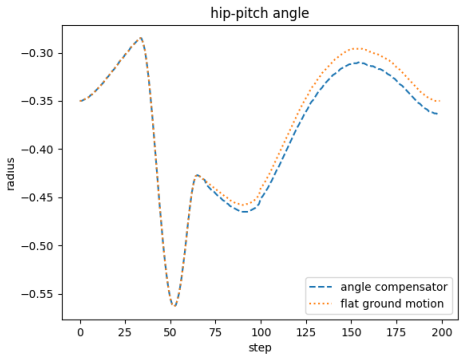


圖 36、不連續地面θ_{R3}變化圖

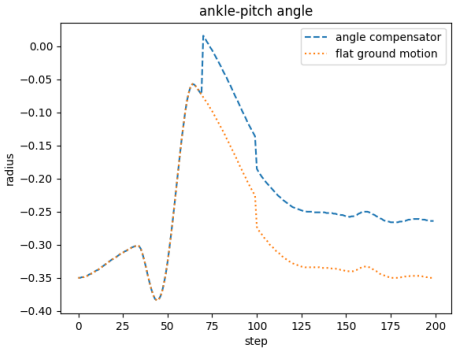


圖 37、不連續地面θ_{R5}變化圖

(2)、斜坡

圖 38 則為不規則坡度的斜坡。當機器人踩上去時，可以看出在 pitch 以及 roll 之間的方向角度受到了衝擊變化，從圖 39 與圖 40 中也可以清楚的觀察到，腳踝的 pitch 跟 roll 方向對此做出了相應的調整，以適應地形的變化。

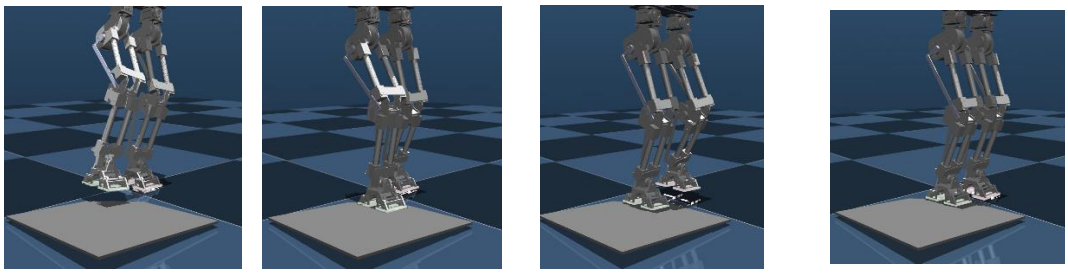


圖 38、斜坡

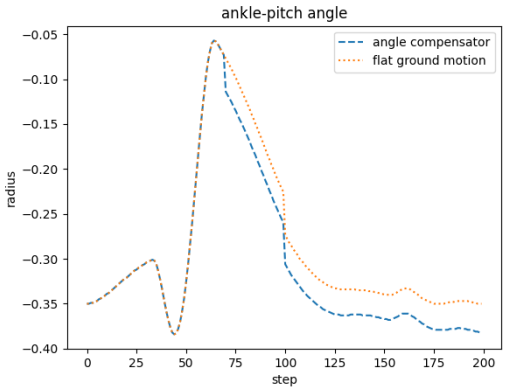


圖 39、斜坡地面θ_{R5}變化圖

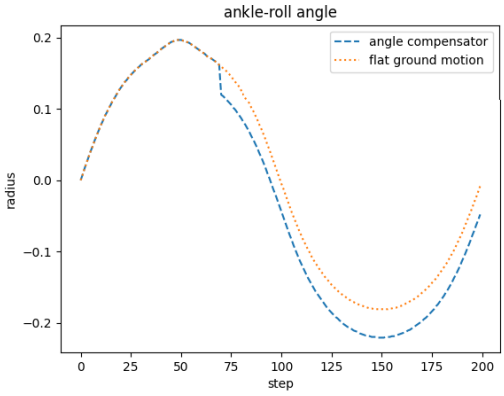


圖 40、斜坡地面θ_{R6}變化圖

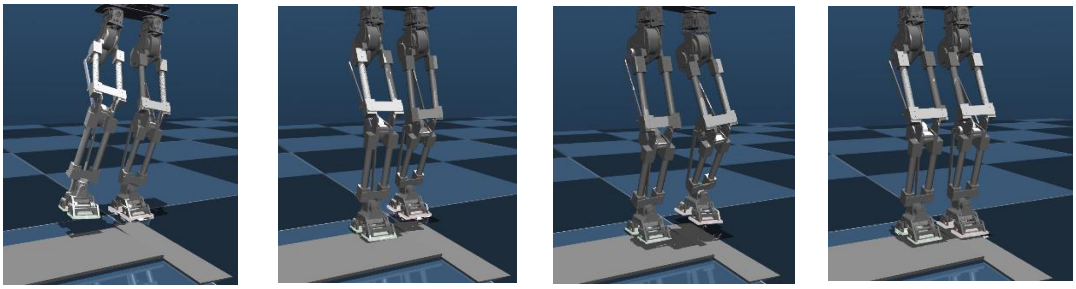


圖 41、多複雜地面

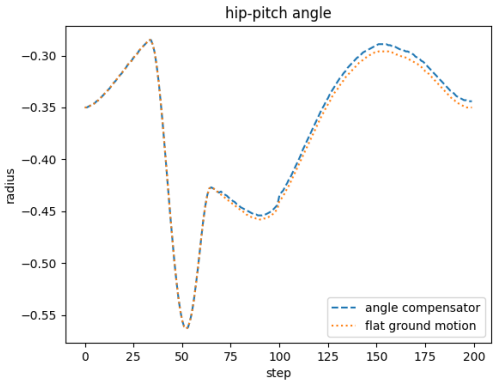


圖 42、多複雜地面θ_{R3}變化圖

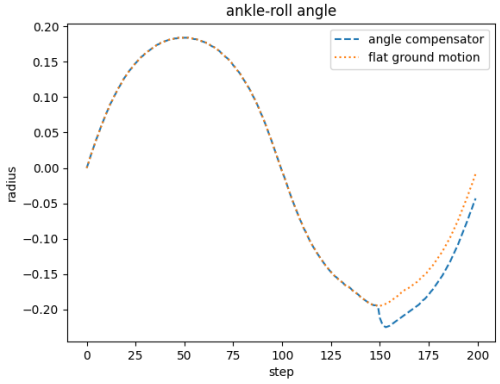


圖 43、多複雜地面θ_{L6}變化圖

(3)、多複雜地面

最後則為左右兩腳將踩上兩種傾斜幅度不一樣的地形，其流程的截圖如圖 41 所示。除了與前面的結果呈現的一樣機器人有針對地形改變腳的擺動軌跡之外，可以看到圖 43 顯示的是右腳腳踝的整體軌跡，由於左腳比較晚接觸到變化的地面，因此角度改變的時間點也比較後面，最後機器人也能成功走上複雜的地面。

五、實體實驗

經由 MuJoCo [15]模擬軟體驗證步態之穩定性及可行性後，本作品設計了實體雙足型機器人的三種實驗場景，一一分述如下:

實驗一、負重行走實現

雙足機器人在倉儲環境的運送任務中，負重對機器人來說是一大挑戰，這裡所謂負重是足以改變機器人重心的重量，並且運送的貨品是無法確認重量的，因此機器人必須要有自適應能力將重心調整回適當位置，以利於接下來的運送任務。這次的運送任務如圖 45，機器人所要運送的貨品為 2 kg 如圖 44，這個重量是為此機器人重量的十分之一，對人來說身體重量的十分之一是個具有負擔的重量。

此次運送任務中，圖 46 至圖 49 前 200 筆數據為機器人從站直到平衡階段，可以看到雙足機器人在被賦予貨品時質心傾斜且零力矩點偏移，通過平衡控制器以及負重策略使機器人在約 350 個時間步長後回到穩定狀態。穩定後機器人開始運用動態的平衡控制器，使其在運動過程中保持穩定，質心角速度的絕對值也幾乎未超過設定的閥值，由此可以看到此控制系統之強健性。

圖 44、貨品重量

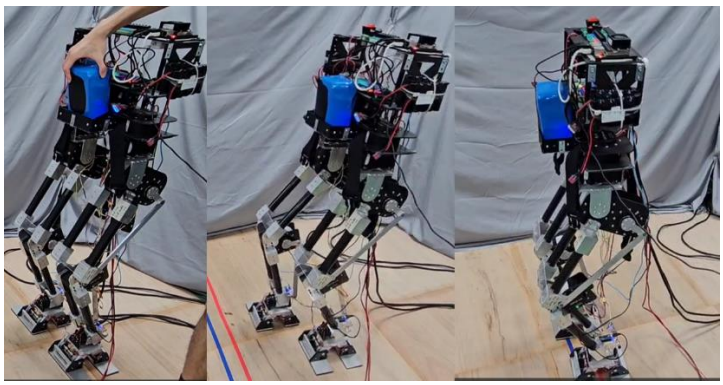


圖 45、運送貨品過程

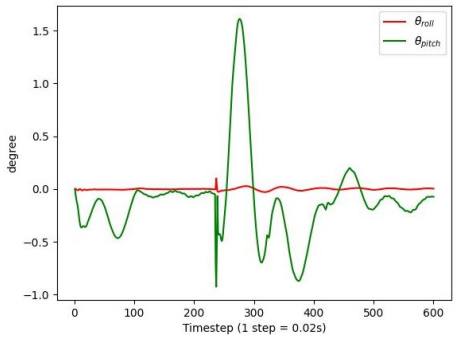


圖 46、質心角度

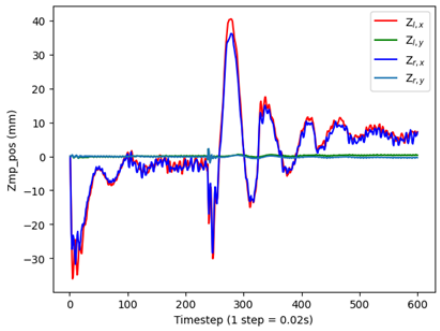


圖 47、零力矩點變化

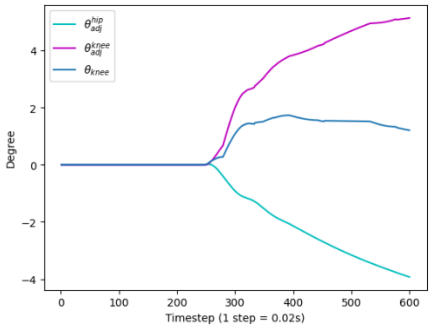


圖 48、負重策略角度變化

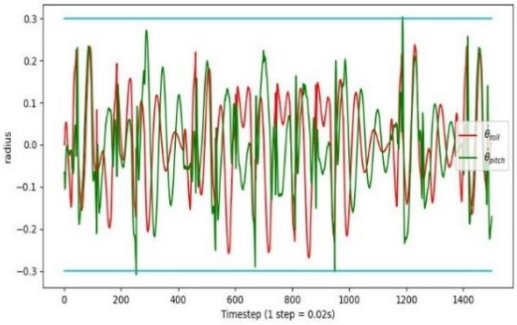
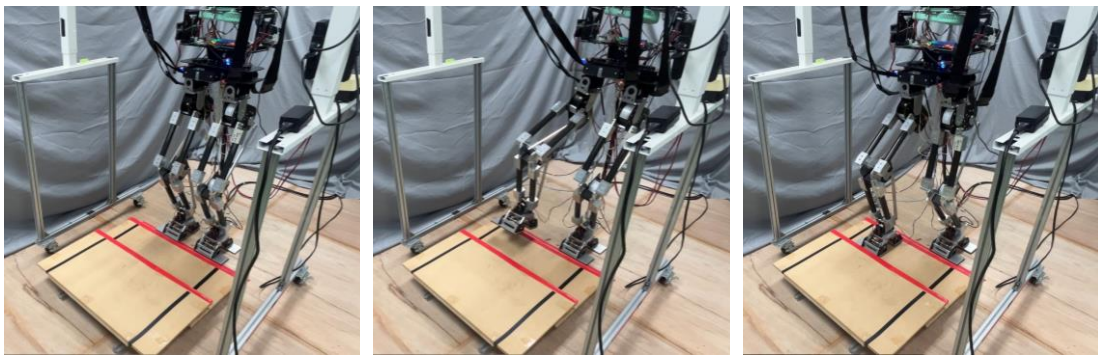


圖 49、質心角速度

實驗二、實體機器人上下樓梯實現

前述在模擬軟體上驗證成功之後，本實驗將這套步態套用至實體機器人之上，並成功完成上下樓梯的任務，其相關過程如圖 50、圖 51 所示。



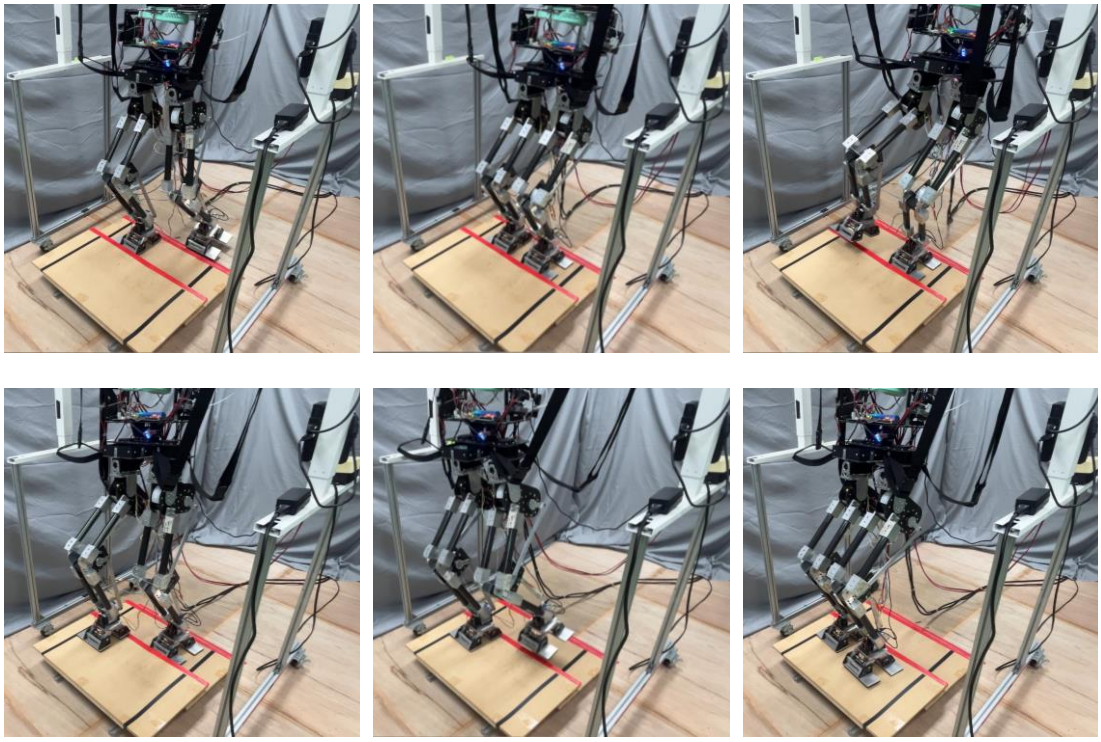
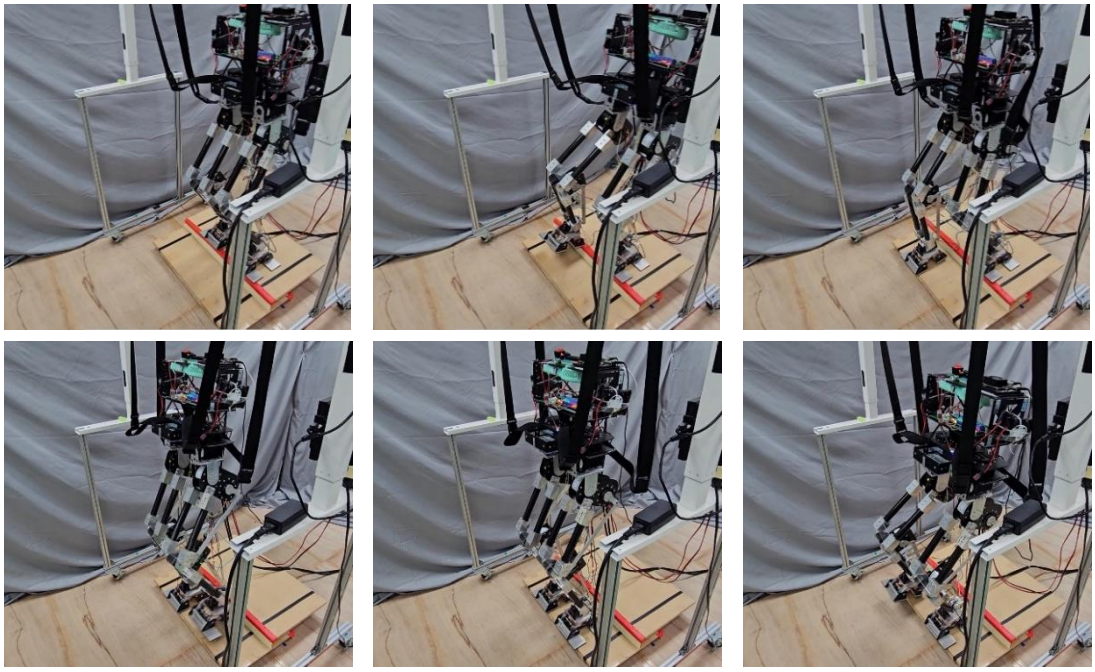


圖 50、實體上樓梯



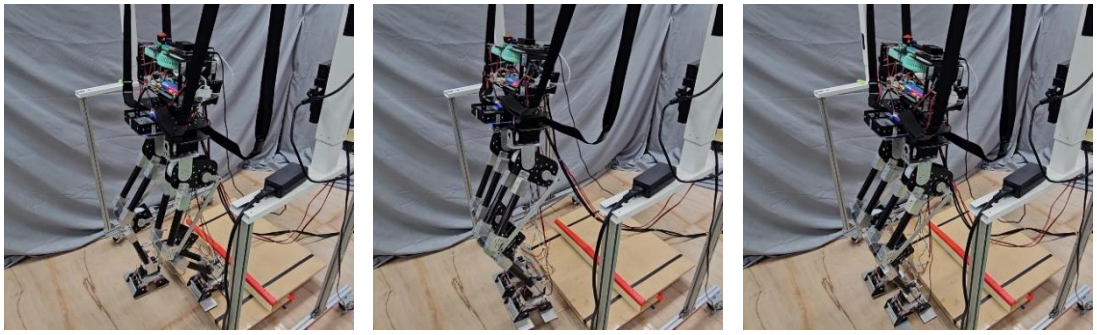


圖 51、實體下樓梯

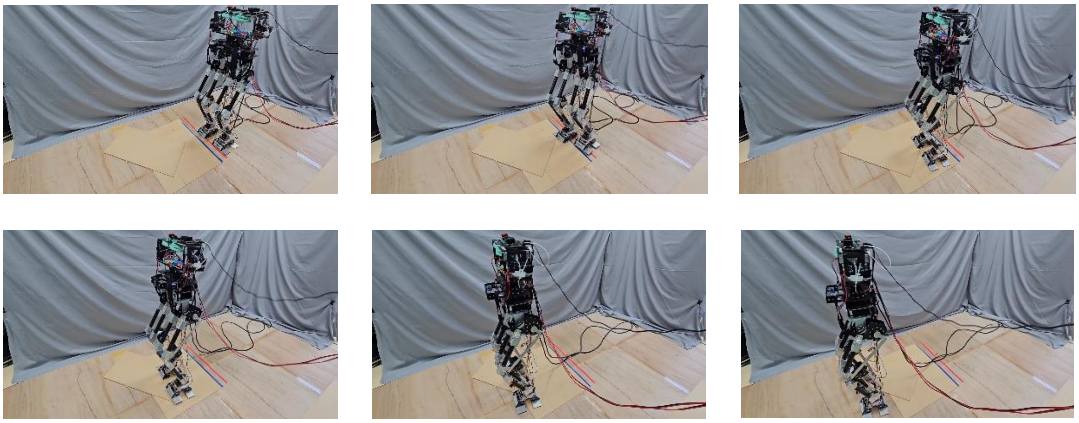


圖 52、不平坦地形行走

實驗三、不平坦地形的橫越

第三部份讓機器人嘗試行走在不規則的地面上。如圖 52 可以看到機器人的前方有數塊木片以不同的方向堆積起來，使得每一步地行走遇到的起伏都會不同。然而機器人能夠透過感知地形的變化去適應並產生出新的符合當前狀況的步行軌跡，最後成功地度過障礙。

七、結論

本企劃實現了人型機器人下半身機構和系統結構的創新設計，以及實體的組裝和測試。腿部設計的目標是實現輕量化，並模仿人類腿部結構，使雙足機器人的質心向上移動，從而減少轉動慣量。其中，我們採用了特殊的四連桿機構設計，使機器人的重心能夠更接近人類。關於腳板的設計，為了減少與環境互動時的振動，該機構模仿了人腳的足弓。此外，在前腳趾和後腳跟部分使用了可旋轉的被動關節。

在操作的層面中，我們讓機器人在行走前能夠先進行平衡，確保一開始的穩定性。除了能夠正常地行走之外，我們的機器人還能更進一步突破許多地形限制，如走樓梯或是不平坦的地形等，使機器人執行任務的層面上能夠更加靈活。

八、參考資料

[1] A. Robotics. “Made for Work, ” 30 Mar., 2024; https://www.youtube.com/watch?v=Jycdks836bY.

[2] B. Dynamics. “ATLAS™ Atlas is a research platform designed to push the limits of whole-body mobility," 31 Mar., 2024; https://bostondynamics.com/atlas/.

[3] Tesla. “Optimus - Gen 2, ” 01 Apr., 2024; https://www.youtube.com/watch?v=cpraXaw7dyc.

[4] M. Xsens. 02 Apr., 2024; https://www.movella.com/products/xsens.

[5] 廣華電子商城．"薄膜壓力傳感器," 02 Apr., 2024; https://shop.cpu.com.tw/product/56021/print/.

[6] J. Kennedy, and R. Eberhart, “Particle swarm optimization. ” pp. 1942-1948 vol.4.

[7] S. Fujimoto, H. Hoof, and D. Meger, “Addressing function approximation error in actor-critic methods. ” pp. 1587-1596.

[8] J. Achiam. “Twin Delayed DDPG, ” 30 Mar., 2024; https://spinningup.openai.com/en/latest/algorithms/td3.html.

[9] D. Manzoni, “Vestibulo-spinal Reflexes, ” *Encyclopedia of Neuroscience*, M. D. Binder, N. Hirokawa and U. Windhorst, eds., pp. 4245-4250, Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[10] T.-H. S. Li, Y.-F. Ho, P.-H. Kuo, Y.-T. Ye, and L.-F. Wu, “Natural Walking Reference Generation Based on Double-Link LIPM Gait Planning Algorithm,” *IEEE Access,* vol. 5, pp. 2459-2469, 2017.

[11] C. Liu, W. Geng, M. Liu, and Q. Chen, “Workspace Trajectory Generation Method for Humanoid Adaptive Walking With Dynamic Motion Primitives,” *IEEE Access,* vol. 8, pp. 54652-54662, 2020.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, “Human-level control through deep reinforcement learning,” *Nature,* vol. 518, no. 7540, pp. 529-33, Feb 26, 2015.

[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.

[14] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, “Deterministic Policy Gradient Algorithms,” *31st International Conference on Machine Learning, ICML 2014,* vol. 1, 06/21, 2014.

[15] E. Todorov, T. Erez, and Y. Tassa, “MuJoCo: A physics engine for model-based control,” in 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2012, pp. 5026-5033.