

國 立 虎 尾 科 技 大 學
機 械 設 計 工 程 系 暨 精 密 機 械 工 程 科
專 題 製 作 報 告

3D 列印模擬在機械手臂設計上的應用

Application of 3D Printing
Simulation in Robot Arm Design

指導教授：嚴家銘老師
班級：四設三甲
學生：
江東祐 (40823116)
江建儒 (40823131)
黃暉翰 (40823152)
蕭日傑 (40823153)

中華民國 111 年 6 月

國立虎尾科技大學
機械設計工程系暨精密機械工程科
學生專題製作合格認可證明

專題製作修習學生：
四設三甲 40823116 江東祐
四設三甲 40823131 江建儒
四設三甲 40823152 黃暉翰
四設三甲 40823153 蕭日傑

專題製作題目：3D 列印模擬在機械手臂設計上的應用
經評量合格，特此證明

評審委員：_____

指導老師：_____

系主任：_____

中華民國一一年六月二十日

摘要

自從 1980 年史上第一個 RP 技術專利的出現後，在時間的推進下，3D 列印之相關技術不斷創新，從最初笨重又昂貴的 3D 列印機，漸漸地輕量化、普及化，價格也愈來愈實惠，時至今日，3D 列印機已經可以應用於多種領域上。

此專題是使用虛擬建構的 3D 列印機，將其導入 CoppeliaSim 模擬環境並給予對應設置，並使用切片軟體對工件進行切片，轉出 Gcode 檔後，運用 GCodeInterpreter 導入 Gcode 檔與 CoppeliaSim 進行對接，操控 CoppeliaSim 場景中的三軸 3D 列印機進行模擬列印，可藉此展示與模擬實際列印的情況。

關鍵字: CoppeliaSim、3D printer、FDM

Abstract

Since the emergence of the first RP technology patent in the history of 1980, under the promotion of time, 3D printing related technologies have been continuously innovated, from the initial bulky and expensive 3D printers, gradually lightweight, popular, and the price has become affordable, and today, 3D printers have been applied in a variety of fields.

This topic is to use a virtually constructed 3D printer, import it into the CoppeliaSim simulation environment and give corresponding settings, and use the slicing software to transfer out the Gcode file, use GCodeInterpreter to import the Gcode file to dock with CoppeliaSim, and control the three-axis 3D printer in the CoppeliaSim scene for analog printing, which can show and simulate the actual printing situation.

Keyword: CoppeliaSim 、 3D printer 、 FDM

誌 謝

在此由衷感謝協助以及製作本專題完成的所有人，首先感謝我們的指導教授嚴家銘教授，一開始專題開始時，我們對於研究的方向都還很迷茫，沒有一個準確的方向，謝謝家銘教授，給予我們的鼓勵與方向上建議，當我們遇到困難時也會不厭其煩地提供我們解決問題的方向和建議，給予我們全方位的支持，在每週開會時也會關心我們的課業狀況，非常感謝老師對我們的關心，在專題上也給了我們能自行探索以及找尋方向的時間，最後感謝大四的學長們對我們的鼎力相助，特此感謝。

目 錄

摘要.....	i
Abstract	ii
誌 謝.....	iii
第一章 前言	1
1.1 研究動機.....	1
1.2 研究目的與方法	1
第二章 現行 3D 列印之技術.....	2
2.1 直角式 FDM 3D 列印機介紹.....	2
第三章 虛擬環境與模擬過程	4
3.1 原理介紹.....	4
3.2 CoppeliaSim	4
3.2.1 使用原因.....	5
3.2.2 RemoteAPI	5
3.2.3 功能列	5
3.3 FIBR3DEmul printer	6
3.3.1 GCodeInterpreter	6
3.3.2 模擬模型.....	7
3.4 模擬流程.....	8
3.4.1 使用 Inventor 繪製 uArm 零件.....	8

3.4.2 使用 Cura 轉出 G-Code 檔	9
3.4.3 更改 G-Code 檔	9
3.4.4 將 G-Code 轉入 GCodeInterpreter	10
3.4.5 uArm 零件模擬列印結果	10
3.4.6 在 CoppeliaSim 匯入 uArm 零件進行模擬	12
3.4.7 簡化模型.....	12
3.4.8 模擬	13
第四章 座標轉換	15
4.0.1 3D 座標提取	15
第五章 pySTL	19
5.1 pySTL 功能.....	19
5.2 為何要使用 pySTL?	20
5.3 如何使用 pySTL?	21
5.4 pySTL 操作設定	21
5.5 程式流程.....	23
5.5.1 pySTL.py 程式流程	23
第六章 電路系統	26
6.1 接線圖.....	26
6.2 Arduino Mega 2560	26
6.3 RAMPS 1.4.....	28
6.4 其餘元件.....	28

第七章 未來研究建議.....	32
第八章 問題與討論	34
參考文獻	36
附錄	38
作者簡介	40

圖 目 錄

圖 3.1 CoppeliaSim Logo	4
圖 3.2 CoppeliaSim 工具列	5
圖 3.3 CoppeliaSim 工具列 (續)	5
圖 3.4 GCodeInterpreter 介面功能介紹	6
圖 3.5 組合圖	7
圖 3.6 使用 Inventor 繪製零件	8
圖 3.7 選擇 Binary 格式	8
圖 3.8 用於轉出 G-Ccode 檔的切片軟體 Cura	9
圖 3.9 模擬中的 GCodeInterpreter 畫面	10
圖 3.10 模擬列印結果	11
圖 3.11 列印選項控制 UI 介面	11
圖 3.12 Toggle shape edit mode(三角形編輯模式)	12
圖 3.13 Convex Hull(凸包)	12
圖 3.14 原版的 uArm 手臂模擬	13
圖 3.15 設計過後的 uArm 手臂模擬	13
圖 3.16 用於控制 3 個馬達軸的 UI 介面	14
圖 4.1 三角網格法線方向	15
圖 4.2 3D 體積的計算	16

圖 5.1 正常轉檔流程	20
圖 5.2 pystl 轉檔流程	21
圖 5.3 clone pySTL	21
圖 5.4 體積與質心數值顯示	22
圖 5.5 uArm scale down coppeliasim	23
圖 5.6 sample.py 程式流程	24
圖 5.7 pySTL.py 程式流程	25
圖 6.1 接線圖	26
圖 6.2 Arduino Mega 2560	27
圖 6.3 RAMPS 1.4	28
圖 6.4 光學限位開關 (optical endstop switch)	29
圖 6.5 步進馬達 (Nema 17 Stepper Motors)	29
圖 6.6 加熱床 (PCB heatbed)	30
圖 6.7 檢製頭 (E3d V6 Hotend)	30
圖 6.8 A4988 步進馬達驅動器 (A4988 Stepper Motor Driver)	31
圖 6.9 電源 12V/20A(Power supply 12V/20A)	31
圖 7.1 未來研究發展	33
圖 8.1 print 錯誤	34
圖 8.2 print 加 ()	34
圖 8.3 self.centroid==None 語法錯誤	34
圖 8.4 將 == 改成 is	35

圖 8.5 self.name 錯誤	35
圖 8.6 將 w 改成 wb	35
圖 8.7 type==b'solid'	36
圖 8 編譯流程	39

表 目 錄

表 3.1 功能說明	6
表 1 文字編輯軟體比較表	38

第一章 前言

1.1 研究動機

全球各種規模的公司都面臨著越來越快節奏、不確定和複雜的邊界條件，驅使這種現象的一個因素是數位化的日益增長迫使公司以更高的成本和時間效率進行開發。這次的專題藉由以客製化以及利於處理複雜的形狀的 3D 列印技術加入數位孿生創建即時的虛擬表示或在物理系統中實現以虛擬描述過程的概念，來達到目的。

1.2 研究目的與方法

第一部分，以 CoppeliaSim 作為虛擬環境，在環境中添加 FIBR3DEuml，並將零件匯入 Cura 轉為 GCode，接著將轉好的 GCode 放進 GCodeInterpreter，模擬列印零件。

第二部分，利用將 pySTL 與 cad 檔做結合達到更精準的結果，將零件轉入虛擬環境模擬進行組配，再進行過程中如有發生錯誤，便可由上游的其他部分進行修改，可有效節省在實體列印上 trial and error 所浪費的時間。最終目的是透過如何利用虛擬環境提高準確性和擬真度以及減少時間、成本和工作量。

第二章 現行 3D 列印之技術

2.1 直角式 FDM 3D 列印機介紹

此專題中使用的 3D 列印機類別為積層製造技術 (Additive Manufacturing,AM) 中的熔融沉積成型 (Fused Deposition Modeling,FDM)，它的列印過程是將線材連續輸送進擠壓頭，透過擠壓頭中的電加熱系統加熱材料，半熔融狀態的材料經過擠壓噴嘴出口時，其線材直徑大約就會等於層厚，經過擠出後材料冷卻定型，再藉由一層層的堆疊而形成三維的立體形狀，因此成品最大的特色為階梯狀的表面。

幾乎所有的積層製造系統一樣，擠製的基礎機器都是採用 STL 格式的 CAD 檔，零件的精度是由精度維持的，故輪廓會以較慢的速度製作，而輪廓是由從 STL 檔的平面和三角形之間提取的交點來做決定，這個方式最突出的優點是在任何想要的座標可以容易的切割。

在成型設備上採用直角座標系統，其工作方式主要是通過完成沿著 X、Y、Z 軸上的線性運動，驅動單元是以伺服馬達或步進馬達為主，以線性滑軌或同步皮帶搭配齒輪或齒條作為傳動元件所架構起來的機械系統。

原形的強度與填充率有關，強度減弱是孔隙率造成的，而孔洞大小是由填料密度去做選擇。

擠製成型的技術結構中，擠製頭大多藉由切層軟體轉換機械指令 GCode 來執行點座標的材料塗層路徑，其中提升 Z 軸升降的穩定性、熔絲擠出量的控制及環境溫度差異變引起的材料熱收縮皆會影響疊層精度的表現，目前都是以手動調整彈簧機構來維持平台角度但此方法無法有效解決平台本身弧形的高低差，而 G92 是透過安裝於效應器上的現為開關偵測，藉由現為開關的觸發機制，以利於傳回平台與每個 Z 軸座標點的高度資

料，當訊號傳回去時，立即儲存該點位置高度，並採線性補差方式來獲得完整的水平成型面。

第三章 虛擬環境與模擬過程

虛擬實境是智慧製造中重要的應用技術之一，利用此技術可以快速驗證產品的製造過程，提前發現潛在的問題，達到縮短產品上市的時間和降低生產成本的目的。

3.1 原理介紹

虛擬機器（英語：virtual machine），在電腦科學中的體系結構裡，是指一種特殊的軟體，可以在電腦平台和終端使用者之間建立一種環境，而終端使用者則是基於虛擬機器這個軟體所建立的環境來操作其它軟體。虛擬機器（VM）是電腦系統的仿真器，通過軟體類比具有完整硬體系統功能的、執行在一個完全隔離環境中的完整電腦系統，能提供物理電腦的功能。

3.2 CoppeliaSim

CoppeliaSim 是一套具有整合開發環境的機器人模擬軟體，基於分佈式控制體系架構，可以利用寫入嵌入式腳本、插件、ROS、BlueZero 節點、RemoteAPI 客戶端或自定義解決方案達成模型控制之效果。



圖. 3.1: CoppeliaSim Logo

並且在 CoppeliaSim 中，控制器可以用 C / C ++、Python、Java、Lua、Matlab 或 Octave 進行編寫。

3.2.1 使用原因

本模擬之最終目標是希望可以在虛擬環境中進行 3D 列印的結果展示，通過虛擬環境中的模擬後，在每次修改零件並更新 Gcode 後可以直接展示列印的狀況，且在虛擬環境中不會有費用的支出，所以可以用於檢視所列印出之成果後，再進行圖檔修正或設計修改，除此之外 CoppeliaSim 的虛擬環境更接近真實環境，基於以上原因，所以此專題選擇 CoppeliaSim 做為模擬的環境。

3.2.2 RemoteAPI

RemoteAPI(Remote Application Programming Interface) 是 CoppeliaSim API 框架的一部分。它允許 CoppeliaSim 與外部應用程序之間的通訊，是跨平臺並支持服務調用和雙向數據流。有兩個不同的版本/框架分別為:Remote API 和 The B0-based remote API。

3.2.3 功能列

1. 以下為簡易功能說明:



圖. 3.2: CoppeliaSim 工具列



圖. 3.3: CoppeliaSim 工具列 (續)

代號	功能說明	代號	功能說明
1	畫面平移	10	複製所有設定
2	畫面旋轉	11	回復/取消回復
3	畫面縮放	12	模擬設定
4	畫面視角	13	開始/暫停/停止模擬
5	畫面縮放至適當大小	14	即時模擬切換
6	選取物件	15	模擬速度控制
7	移動物件	16	線程渲染/視覺化
8	旋轉物件	17	場景/頁面選擇
9	加入/移出樹狀結構		

表. 3.1: 功能說明

3.3 FIBR3DEmul printer

FIBR3DEmul--是一個適用於 3 軸或 3 軸以上的 FDM(熔融沉積成型)列印機，進行虛擬列印的開源軟體，在它之中包含了 GCodeInterpreter(GCode 解析器)、使用 CMake 生成的 simExtFIBR3D.dll

3.3.1 GCodeInterpreter

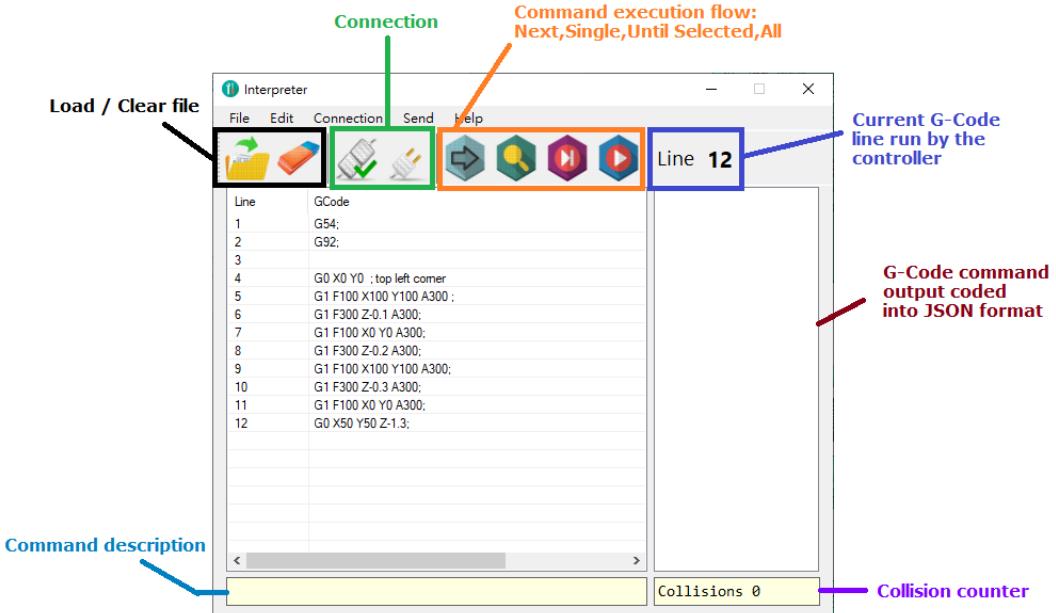


圖. 3.4: GCodeInterpreter 介面功能介紹

3.3.2 模擬模型

在模擬的模型上，延用了學長之前組建的實體 3D 列印機，並將其轉為虛擬模型後放入 CoppeliaSim，進行組裝、配置後以 simExtFIBR3D.dll 擴充 CoppeliaSim 將其與 GCodeInterpreter 程式進行整合，用以達成使用 GCode 控制 CoppeliaSim 中的 3D 列印機進行模擬列印展示之功能。

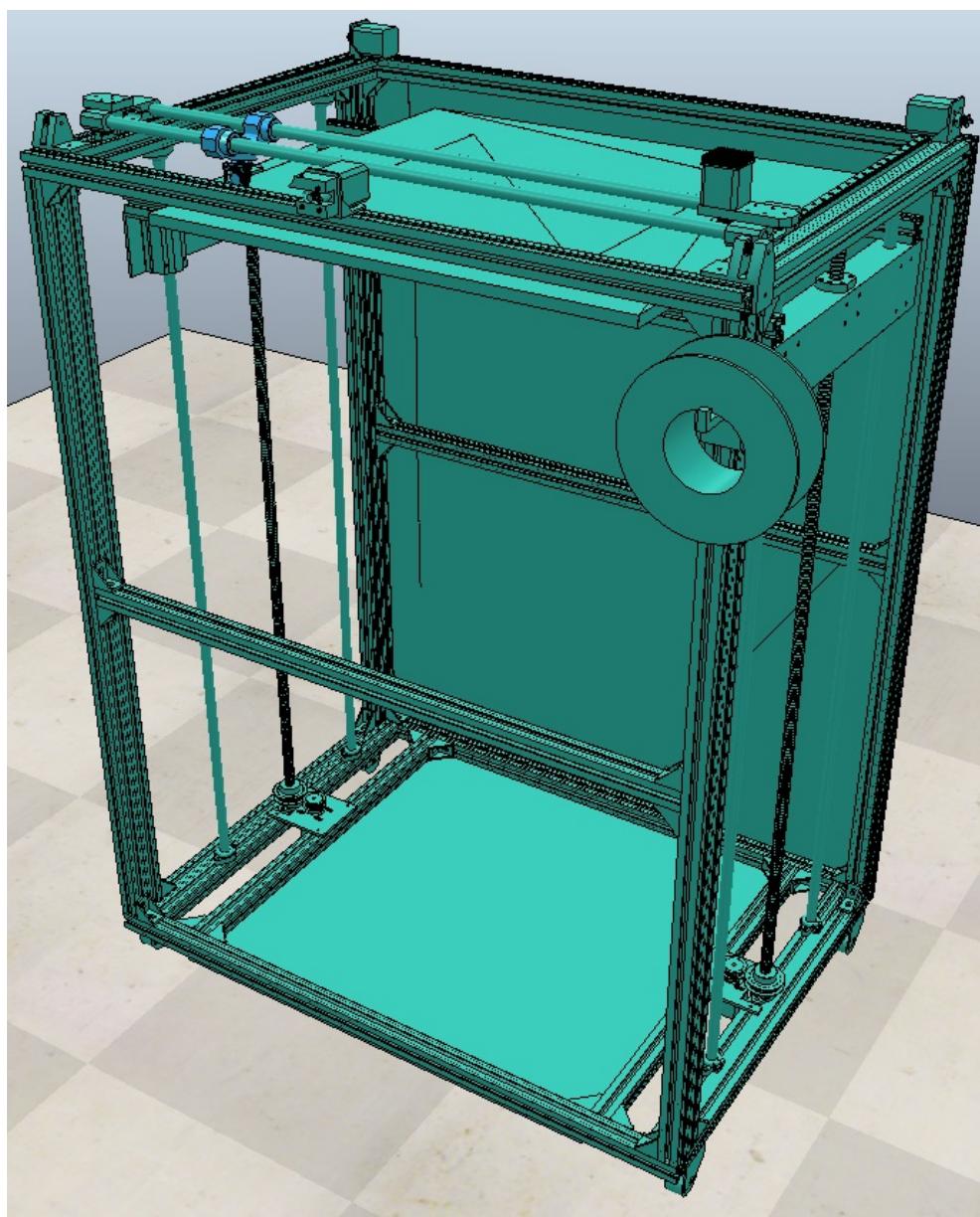


圖. 3.5: 組合圖

3.4 模擬流程

3.4.1 使用 Inventor 繪製 uArm 零件

模擬的第一步是先將更改後的 uArm 零件使用 Inventor 繪製出來，因為 Cura 能夠辨識出 STL、OBJ 或是 3MF 這些文件格式，所以這裡將檔案轉為 STL 的 Binary 格式，之後使用 Cura 將 STL 做切片並且產生 uArm 零件的 G-code 程式。

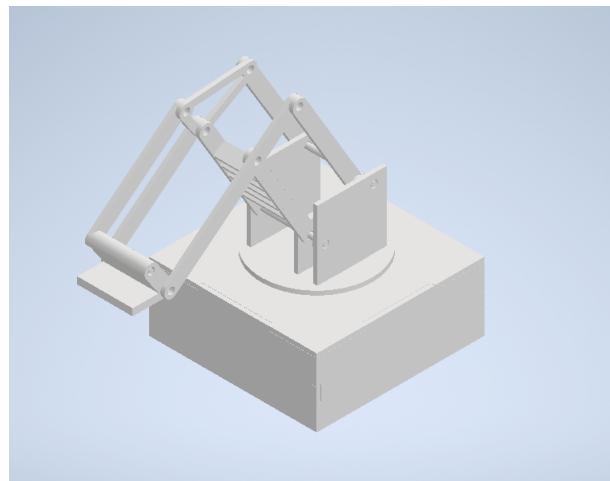


圖. 3.6: 使用 Inventor 繪製零件

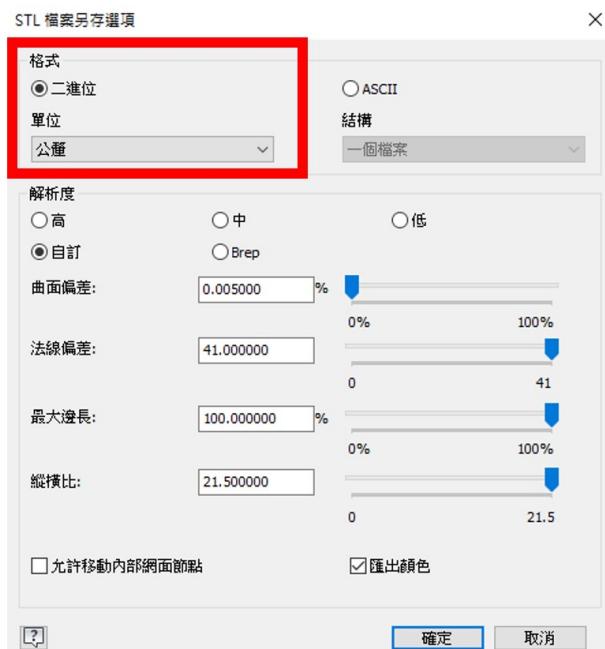


圖. 3.7: 選擇 Binary 格式

3.4.2 使用 Cura 轉出 G-Code 檔

第二步將把上述步驟中的 STL 檔案匯入到 Cura 程式中，Cura 可以將 STL 檔進行矩陣排列或旋轉方向，並將 STL 檔放置到最佳的列印位置後，使用 Cura 將 uArm 進行切片，把 uArm 的 STL 格式轉換成的 G-Code 檔案，這裡因為要讓 3D Print 中的控制器與伺服馬達產生溝同橋梁的程式代碼 G-code，所以要將 STL 檔轉換成機器看得懂的 G-code 格式，這裡的 G-code 是要由切層軟體 Cura 生成。

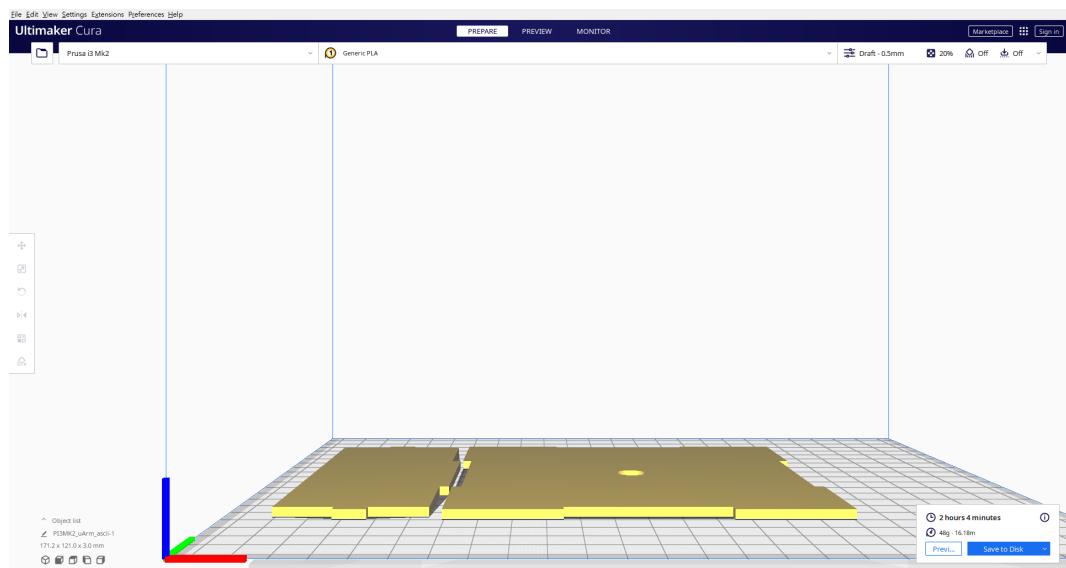


圖. 3.8: 用於轉出 G-Code 檔的切片軟體 Cura

3.4.3 更改 G-Code 檔

由於此列印模組之 Z 軸與本專題之 3D 列印機恰好相反，因此需修改有關的 Z 軸代碼後的數值為負值，還有一處需要修改，那就是擠出率的代碼，列印模組中擠出率代碼為 A，但從 Cura 中轉出的 G-Code 中擠出率代碼為 E，所以需將 G-Code 中的代碼 E 更改為 A。

3.4.4 將 G-Code 轉入 GCodeInterpreter

第四步將修改好的 G-Code 檔案轉入 GCodeInterpreter，按下連接鍵與 CoppeliaSim 連接後，進行模擬。

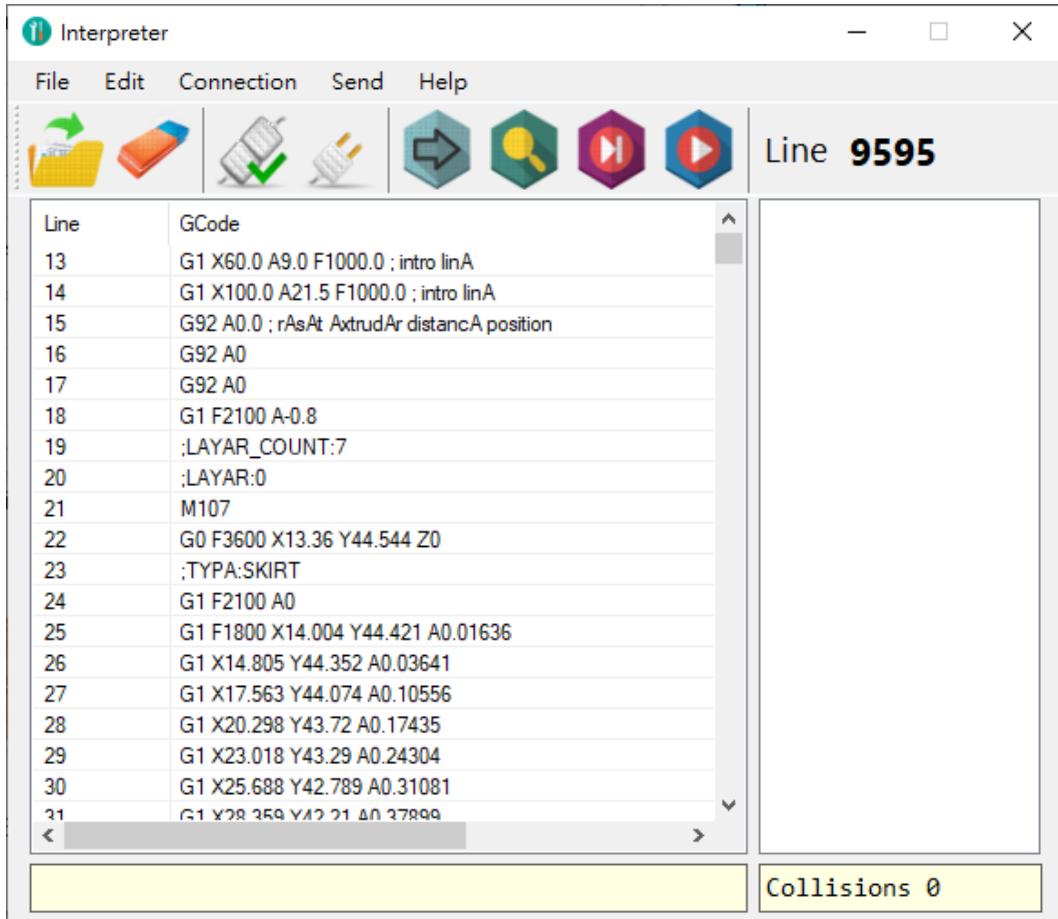


圖. 3.9: 模擬中的 GCodeInterpreter 畫面

3.4.5 uArm 零件模擬列印結果

在 CoppeliaSim 模擬設定視窗中可以調整擠出材料的大小以及形狀(圓球或方塊)，與 GCodeInterpreter 連接後，當 GCodeInterpreter 按下運行鍵後，開始列印模擬，在 CoppeliaSim 中也可以進行模擬列印整體速度調整，更加省時，下圖為模擬結果。

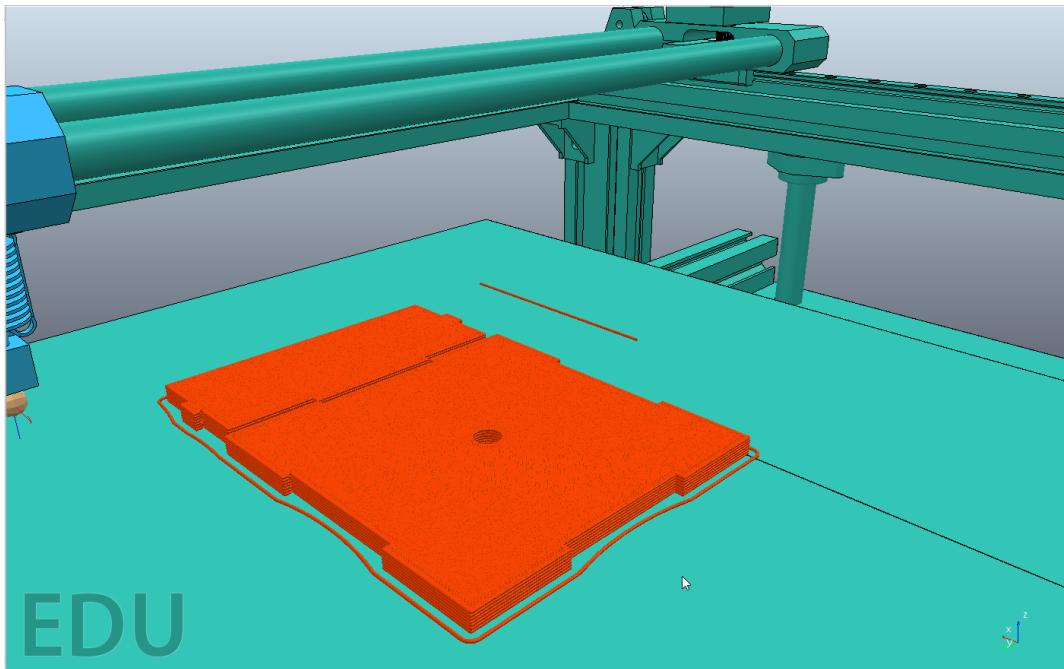


圖. 3.10: 模擬列印結果

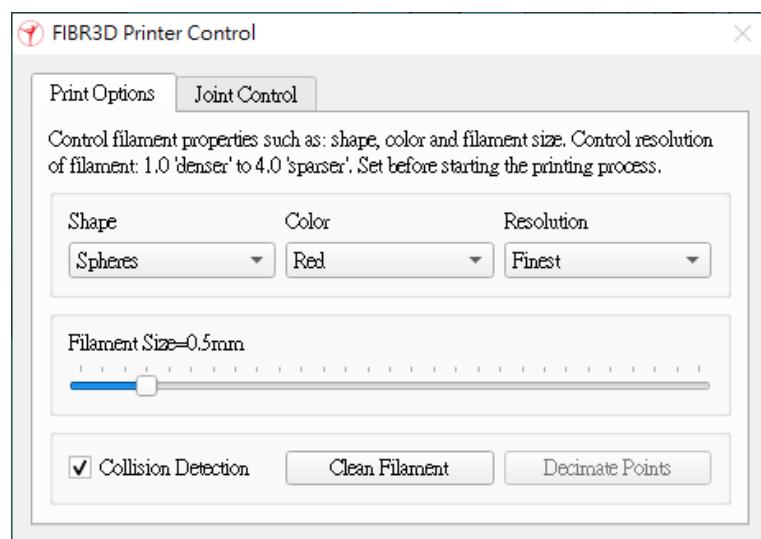


圖. 3.11: 列印選項控制 UI 介面

3.4.6 在 CoppeliaSim 匯入 uArm 零件進行模擬

將 uArm 的 STL 檔使用 pySTL 進行比例縮放後，匯入 CoppeliaSim 中將零件進行拆解並簡化，接著是各軸的組配，最後搭配 CoppeliaSim 的 UI 進行三個馬達軸的控制。

3.4.7 簡化模型

所謂的簡化係指將 uArm STL 檔透過 pySTL 進行縮放，轉進 CoppeliaSim 拆解後，進入 Toggle shape edit mode(三角形編輯模式)針對各零件的三角網格進行選擇後，從複雜形狀的零件簡化提取出方塊、圓柱或圓球等等的簡單形狀，用於模擬運算，外觀可套上原零件保持原樣，另一方式為使用 add 選單中的 Convex decomposition of selection(凸分解)，Convex Hull(凸包)可以理解在高維空間中有一群散佈各處的點，「凸包」是包覆這群點的所有外殼當中，表面積暨容積最小的一個外殼，而最小的外殼一定是凸的。而凸分解就是將零件依據三角網格的頂點分解成數個凸包再進行組合，在運算效率上來說使用 Toggle shape edit mode(三角形編輯模式)方式可獲得較好的效果。

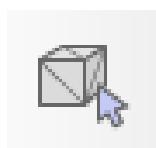


圖. 3.12: Toggle shape edit mode(三角形編輯模式)

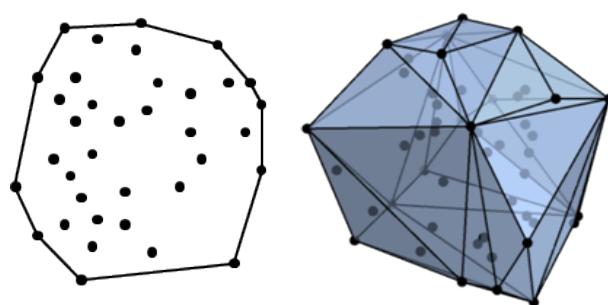


圖. 3.13: Convex Hull(凸包)

3.4.8 模擬

一開始使用官方圖檔進行模擬，花費許多組配時間在簡化零件之上，所以決定使用自行繪製的簡化圖檔進行模擬，簡化圖檔保留了主要的零件，設計後的零件大部分為利於列印的薄板件。

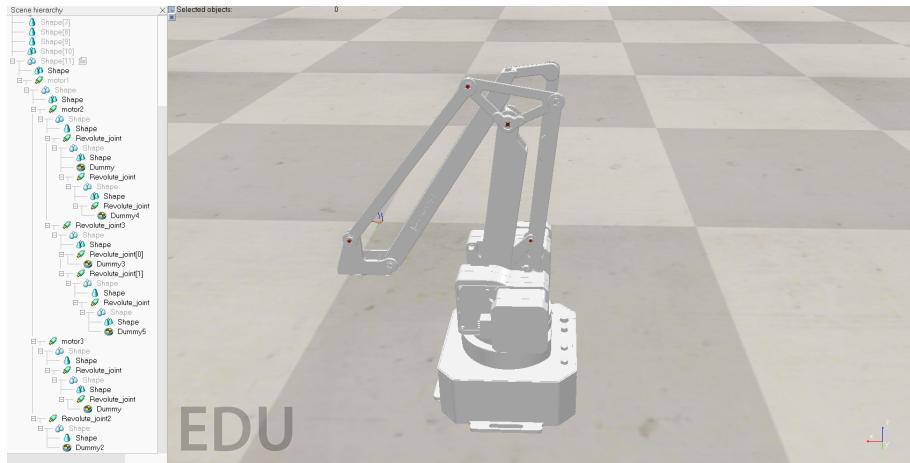


圖. 3.14: 原版的 uArm 手臂模擬

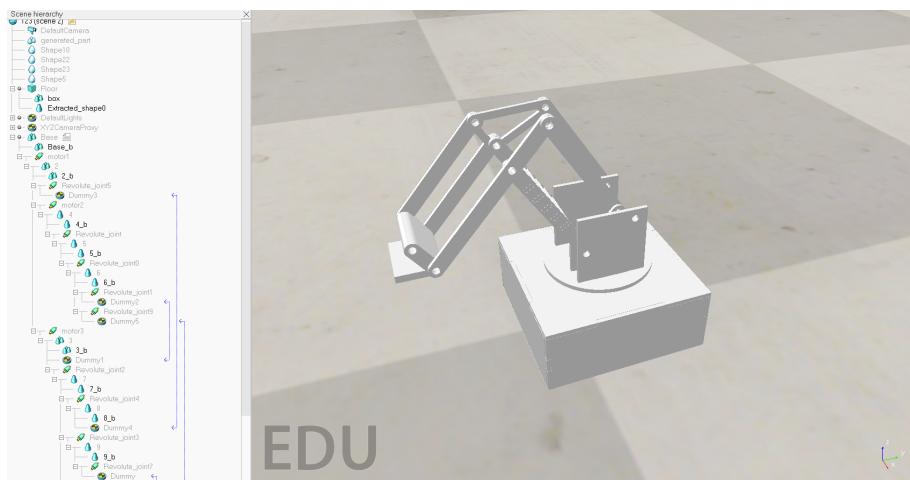


圖. 3.15: 設計過後的 uArm 手臂模擬

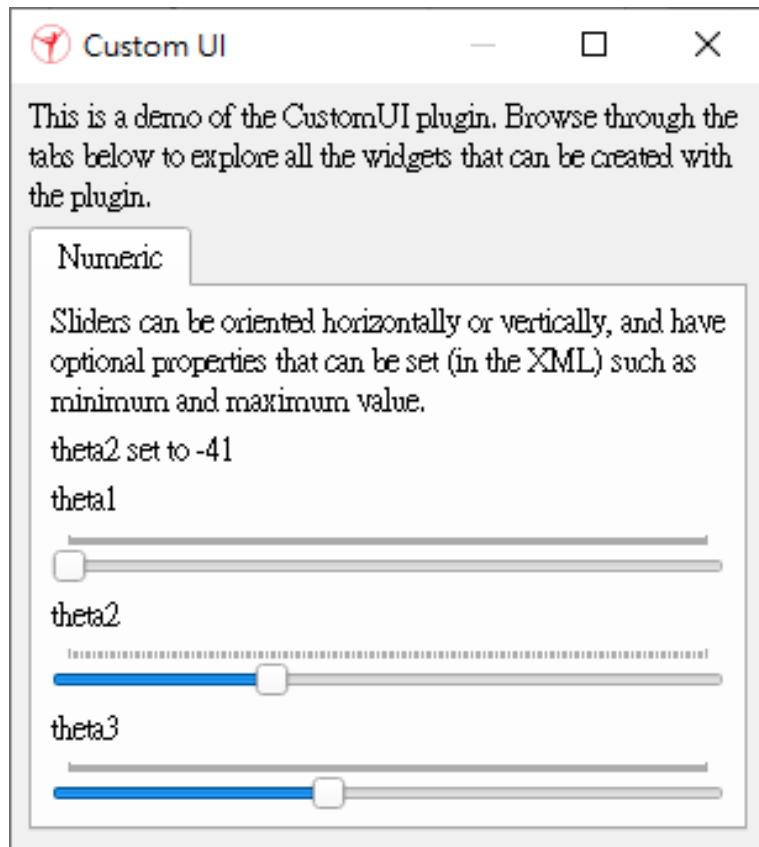


圖. 3.16: 用於控制 3 個馬達軸的 UI 介面

第四章 座標轉換

4.0.1 3D 座標提取

座標轉換是為了將複雜積分或傅立葉級數轉換為簡單直觀的公式，讓他變成最基本的三角形或四面體網格去計算，再將所有網格的值相加，得到體積和慣性矩後求得質心。

- 網格及法線的形成(圖.4.1)：

STL文件表示的表面是封閉並連接的三角形網格，其中每條邊都是兩個三角形的一部分，並且不相交。對於一個三角形，可以通過頂點的順序和右手定則來決定，如果公共邊有不同的方向，就可以證明兩個三角形的法線是一致的。

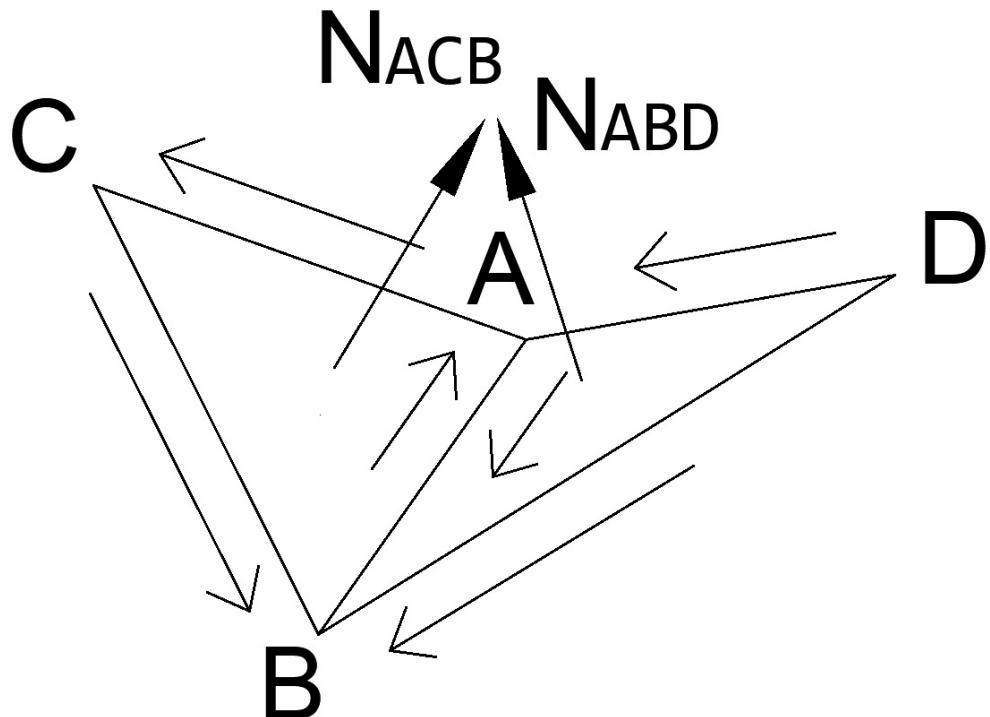


圖. 4.1: 三角網格法線方向

- 四面體體積 (圖.4.2) :

在 3D 情況下，計算的基本單元為四面體，將原點與三角形的各個頂點連接形成一個四面體。三角形 ACB 具有法線 \mathbf{N}_{CAB} ，由於原點 O 位於 \mathbf{N}_{CAB} 的對面，因此這個四面體的值是正的，體積也可以由內積 $\mathbf{OA} \cdot \mathbf{N}_{CAB}$ 判斷。四面體體積為：

$$V'_t = \frac{1}{6}(-x_3iy_2iz_1i + x_2iy_3iz_1i + x_3iy_1iz_2i - x_1iy_3iz_2i - x_2iy_1iz_3i + x_1iy_2iz_3i)$$

$$V'_t \text{otal} = \sum_i V'_i$$

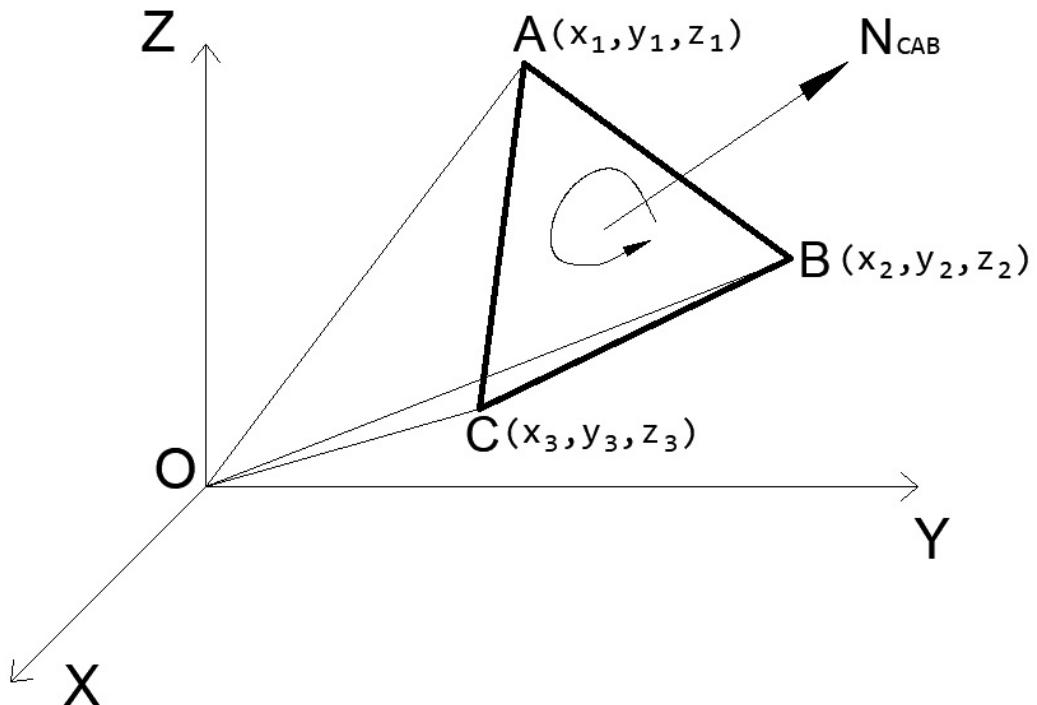


圖. 4.2: 3D 體積的計算

- 空間裡的體積：

計算的特徵可以寫為帶有座標總和的基礎特徵形狀，並且可以以顯式導出基本形狀，雖然這是一個很大的約束，但是在內部空間上具有積分形式都可以使用。

p, q, r 為矩的階，中心矩可以從下列求解：

$$M_{pqr} = \iiint x^p y^q z^r \rho(x, y, z) dx dy dz$$

其中 $\rho(x, y, z)$ 是基礎形狀 i 的指示函數：

$$\rho(x, y, z) = \begin{cases} 1, & \text{if } (x, y, z) \text{ is inside the mesh} \\ 0, & \text{otherwise} \end{cases} \quad (4.1)$$

S_i 是形狀 i 做有標記座標的體積的符號函數，積分可以重寫為每個基本形狀的積分之和：

$$M_{pqr} = \sum_i S_i \iiint x^p y^q z^r \rho_i(x, y, z) dx dy dz$$

由於物體內部的空間可以使用傅立葉變換，也能通過將積分分解為每個基本形狀的積分來計算。二維的傅里葉轉換或 3D 網格模型由傅里葉變換定義其指示函數：

$$\Theta(u, v, w) = \iiint e^{-i(xu+yv+zw)} \rho(x, y, z) dx dy dz$$

- 產生矩陣：

主軸是通過計算矩陣 S 的特徵向量獲得的，也稱為主成分分析 (PCA)。將最大特徵值對應的特徵向量作為第一主軸。第二個特徵值對應的下一個特徵向量是第二個主軸，以此類推。我們進一步確保三階矩 M300 和 M030 變換後是正的，為了使最終結果是唯一的。

我們通過 3D 模型的二階矩構造一個 3x3 矩陣：

$$S = \begin{bmatrix} M_{200} & M_{110} & M_{101} \\ M_{110} & M_{020} & M_{011} \\ M_{101} & M_{011} & M_{022} \end{bmatrix}$$

- 總結：

要使用 pySTL 將 STL 檔在轉入虛擬環境時，坐標軸會產生偏差，需要用到的體積和慣性矩去轉換質心，利用質點繞了座標軸的概念矯正零件的偏差，即可調整物體的角度、移動與比例縮放：

M_{000} 數字符號由左至右分別代表著 X、Y、Z，而數字大小則代表冪次數。

體積： $M_{000} = \frac{1}{6}(-x_3y_2z_1 + x_2y_3z_1 + x_3y_1z_2 - x_1y_3z_2 - x_2y_1z_3 + x_1y_2z_3)$

對 x 的一次矩： $M_{100} = \frac{1}{4}(x_1 + x_2 + x_3)M_{000}$

對 x 的二次矩： $M_{200} = \frac{1}{10}(x_1^2 + x_2^2 + x_3^2 + x_1x_2 + x_2x_3 + x_1x_3)M_{000}$

質點 x 軸位置： $\frac{M_{100}}{M_{000}} = (\frac{1}{4}(x_1 + x_2 + x_3))$

質點 y 軸位置： $\frac{M_{010}}{M_{000}} = (\frac{1}{4}(y_1 + y_2 + y_3))$

質點 z 軸位置： $\frac{M_{001}}{M_{000}} = (\frac{1}{4}(z_1 + z_2 + z_3))$

第五章 pySTL

pySTL 是處理 stl 檔案的 python 開源程式，無須打開 CAD 軟體即可編輯操作 stl 檔案，可以輕易使物體移動、轉動與縮放的動作。目前可以把輸入的 ASCLL 或 binary STL 檔案全部輸出成 ASCLL STL。

5.1 pySTL 功能

- 打開 text.stl 檔案

程式. 5.1: 輸入檔案

```
import pySTL
model = pySTL.STLmodel('text.stl')
```

- 物體向 X 軸移動 10 單位

程式. 5.2: 移動

```
import numpy
movement = numpy.array([10, 0, 0])
model.translate(movement)
```

- 物體質心移動到座標原點

程式. 5.3: 移動

```
model.translate(-c)
```

- 物體向 X 軸轉 90 度 (角度是徑度)

程式. 5.4: 轉動

```
R= pySTL.rotationAboutX(-3.14149/2)
model.rotate(R)
```

- 物體縮小 10 倍

程式. 5.5: 縮放

```
scale = 0.1
model.scale(scale)
```

- 質心顯示

程式. 5.6: 數值

```
c = model.get_centroid()
print(c)
```

- 體積顯示

程式. 5.7: 數值

```
v = model.get_volume()
print(v)
```

- 產生 newText.stl 檔案

程式. 5.8: 產生新 stl 檔案

```
model.write_text_stl('newText.stl')
```

5.2 為何要使用 pySTL?

由於預設場景裡 CAD 檔案與.STL 的座標系是不相同，且.stl(mm) 與 coppeliasim(m) 的單位也不相同，所以當人們在不同軟體間操作，若是沒有注意到各個檔案的關係，會發生(如：圖 5.1)物體方向錯誤或者尺寸錯誤，則必須要打開 CAD 軟體，使用軟體的移動、旋轉、縮放等工具把問題修正再轉成 stl 格式放進 CoppeliaSim 場景，這一來一往所花費的時間是耗時的。

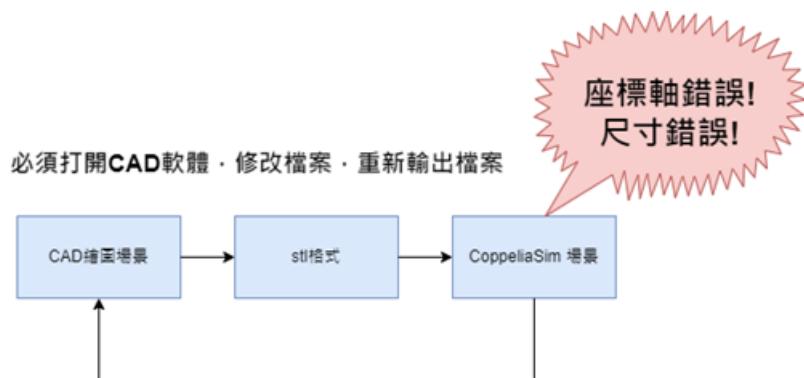


圖. 5.1: 正常轉檔流程

當檔案錯誤時，無須打開 CAD 軟體，只要編輯器啟動 pySTL 代碼，輸入想要修改的參數按下執行，即可產生正確且最新版本的 stl 檔案，有了這項工具，即使電腦裡沒有繪圖軟體，也能立刻修改出正確的檔案，這將會是簡單又省時的操作來完成目標。(如：圖 5.2)

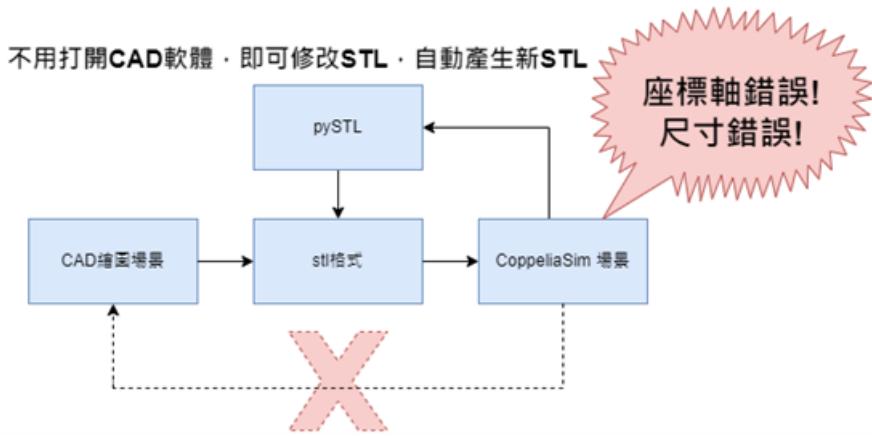


圖. 5.2: pySTL 轉檔流程

5.3 如何使用 pySTL?

因為 pySTL 是一個開源套件，所以我們先在 github 裡 git clone pySTL 整個資料夾下載出來。(如: 圖 5.3)



圖. 5.3: clone pySTL

這裡我們先當作檔案已經排除錯誤可以直接使用，因為資料裡的 pySTL.py 與 sample.py 有多處地方是錯誤的需要去解決，我們將詳細的解決步驟放到第 8 章節的問題討論裡。

5.4 pySTL 操作設定

首先打開 sample.py 檔案，輸入想要操作的名稱的 STL 檔。

程式. 5.9: 輸入名稱

```

import pySTL
from numpy import array
#Load a model from a file.
model = pySTL.STLModel('uArm_binary.stl')

```

在這裡我們選擇的檔案是 uArm_binary.stl 檔案，並且可以先使用 print，使我們可以先取得原本 stl 檔案的體積與質心數值。(圖:5.4)

程式. 5.10: 數值

```
v = model.get_volume()  
print(v)
```

	Attempting to read binary STL: uArm_binary.stl Volume 565028.9126444063 Centroid X: 56.51707121557581 Y:95.25823560123528 Z:-230.1726044013105	
--	--	--

圖. 5.4: 體積與質心數值顯示

我們輸入 pySTL.rotationAboutX(-3.14159/2)，使物體能夠向 x 軸轉動 90 度，這裡需要注意的點是，角度的單位是徑度。

程式. 5.11: 向 X 軸轉 90 度

```
#Rotate the model 90 degrees about the X-axis  
R2 = pySTL.rotationAboutX(-3.14159/2)  
model.rotate(R2)  
c = model.get_centroid()  
print ("Centroid1 " + "X: " + str(c[0]) + " Y: " + str(c[1]) +  
      " Z: " + str(c[2]))
```

因為 stl 檔案的單位 mm，但是 coppeliasim 模擬的場景的單位是 m，所以我們必須使用 scale=0.001，使物體縮小 1000 倍在使用 model.write_text_stl 來產生檔案。(如圖: 圖 5.5)

程式. 5.12: 縮小 1000 倍

```
#Scale the model down by 1000%  
scale = 0.001  
model.scale(scale)  
c = model.get_centroid()  
print ("Centroid2 " + "X: " + str(c[0]) + " Y: " + str(c[1]) +  
      " Z: " + str(c[2]))  
model.write_text_stl('uArm_scale_down_0.001.stl')
```

最後我們把生成好的 uArm_scale_down_0.001.stl 放進 coppeliasim 場景裡面(圖 5.4)我們可以發現 uArm 手臂的尺寸與方向都是正確的，無須再調整。



圖. 5.5: uArm scale down coppeliasim

5.5 程式流程

首先我們執行 sample.py 的啟動檔，導入 pySTL 模組，即可讀取 STL 檔案，得知物體體積與質心數值後，輸入想要的物體位置與比例縮放，來達成最終理想的 STL 檔案。(圖:5.6)

5.5.1 pySTL.py 程式流程

(圖:5.7) 我們讀取 STL 會先判斷各個種類的 STL 檔案，分別使用不同的方法去讀取檔案，當可以成功讀取檔案時，我們即可以使用 ASCII 的規格來寫入進去 (solid-表面對法線-三維三角形頂點)，即可產生 ASCII STL 檔案。

我們使用每個三角形的頂點與原點形成一個四面體，計算每個模型體積用於得到三維形狀的質心 (使用第 6 章的理論來求解)，即可得用於物體的移動、轉動與縮放。

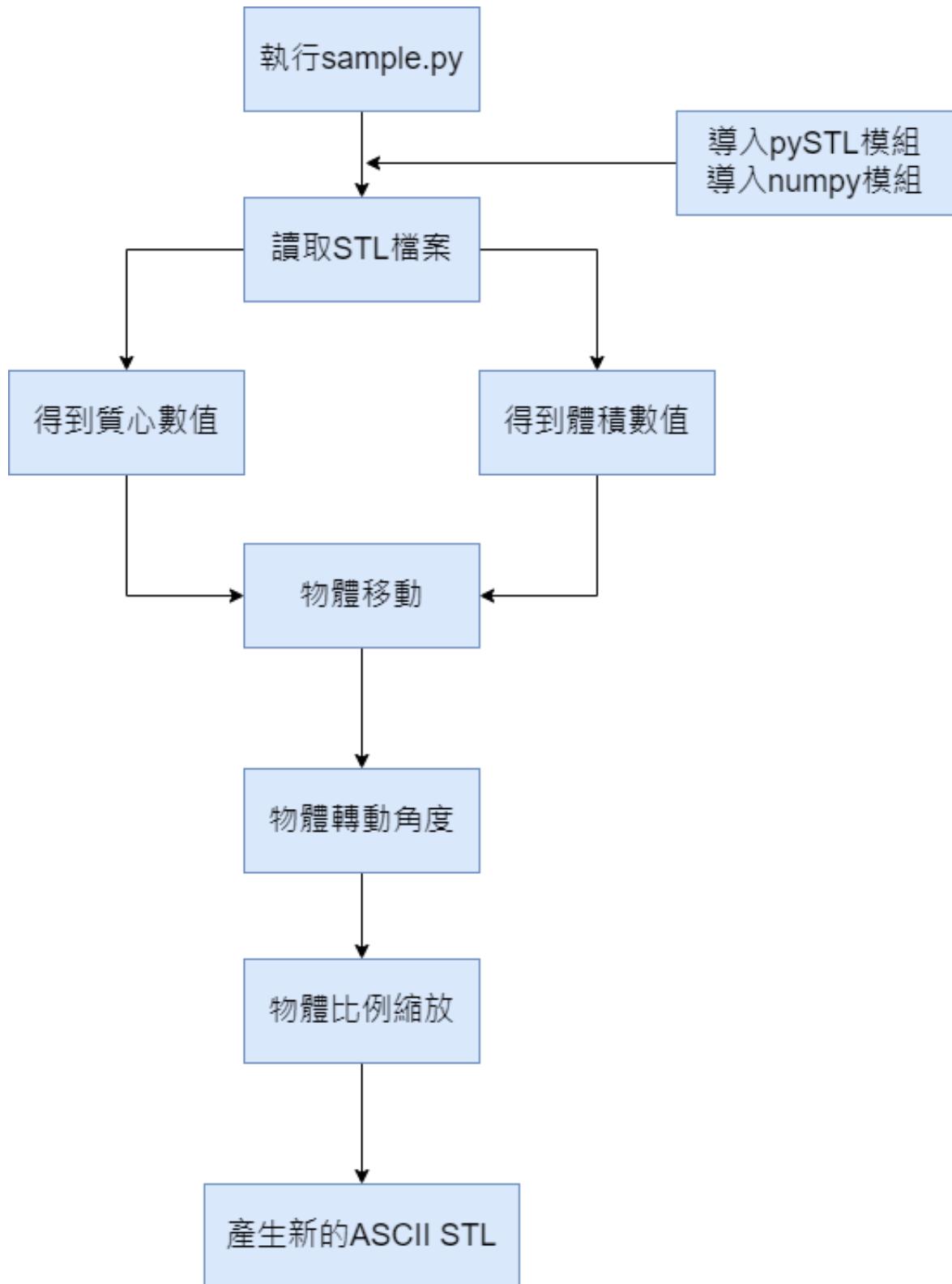


圖. 5.6: `sample.py` 程式流程

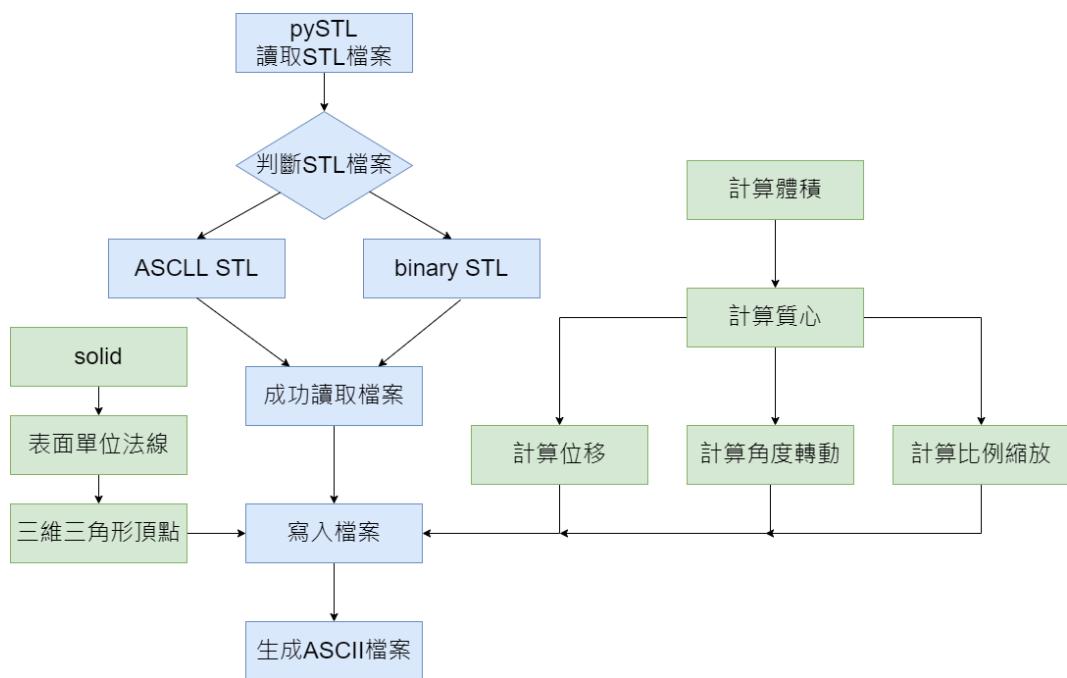


圖. 5.7: pySTL.py 程式流程

第六章 電路系統

6.1 接線圖

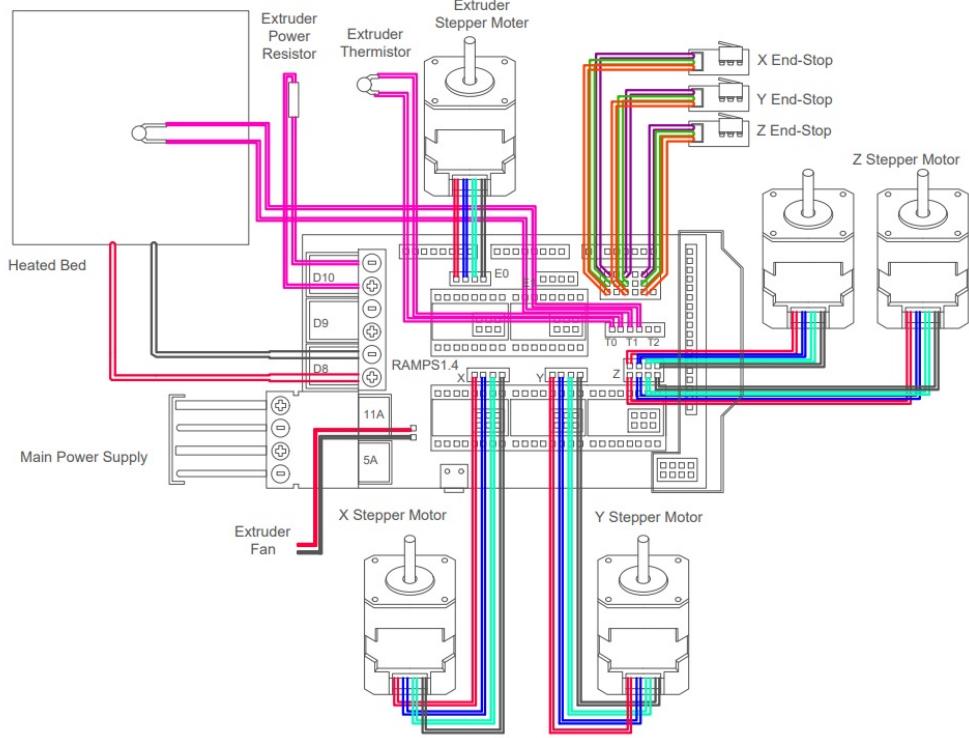


圖. 6.1: 接線圖

6.2 Arduino Mega 2560

Arduino Mega 2560 是基於 ATmega2560 的主控開發板。Arduino Mega 2560 是採用 USB 接口的核心電路板。它具有 54 個數位輸入輸出，適合需要大量複雜 I/O 接口的設計。核心處理器為 ATmega2560，有內建 Pull-Up 電阻在 IC 內部，不需要在外部電路另外安排 Pull-Up 電阻，同時具有 54 路數位輸入/輸出口、16 個類比輸入，4 個 UART 接口、1 個 16MHz 晶體振盪器、1 個 USB 連接口、1 個電源插孔、1 個 ICSP 接頭以及 1 個復位按鈕。Arduino Mega 2560 也能兼容 Arduino NUO 設計的擴展板。可以自動選擇 3 中供電方式：外部直流電源通過電源插座供電；電池連接電源連接器的 GND 和 VIN 引腳；USB 接口直流供電，整合了微控制器以及燒錄功能於

一身。

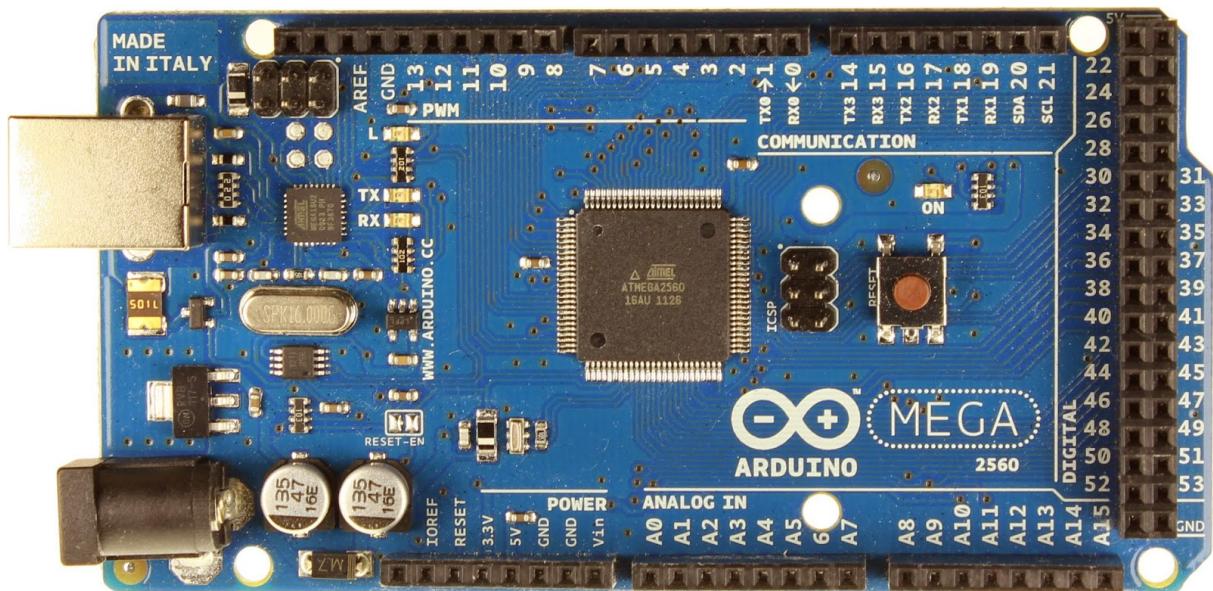


圖. 6.2: Arduino Mega 2560

6.3 RAMPS 1.4

RAMPS 1.4 為 RepRap Arduino Mega Pololu Shield 的縮寫，主要為設計給步進馬達驅動器的介面電路，1.4 為電路版本號碼，目前最新版本為 1.7，Arduino Mega 2560 可透過 RAMPS 1.4 介面與控制電路溝通進而達成控制步進馬達以及其他硬體的功能，目前可控制 X、Y、Z 三軸與最多兩個擠出頭以及兩個散熱風扇。

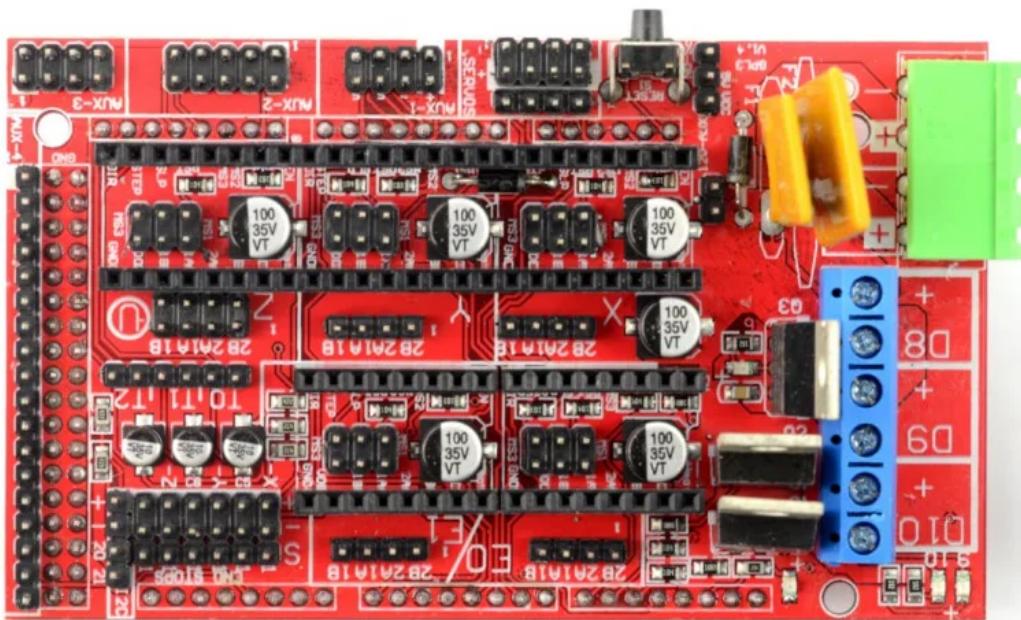


圖 . 6.3: RAMPS 1.4

6.4 其餘元件

光學限位開關的連接需要用到 3 條線，分別接到 RAMPS 上的 (S)、(-) 及 (+)，這 3 個腳位。



圖. 6.4: 光學限位開關 (optical endstop switch)

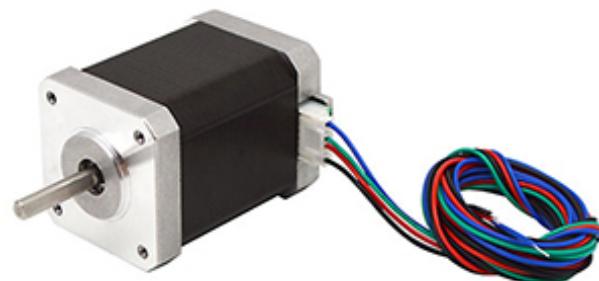


圖. 6.5: 步進馬達 (Nema 17 Stepper Motors)



圖. 6.6: 加熱床 (PCB heatbed)



圖. 6.7: 擠製頭 (E3d V6 Hotend)

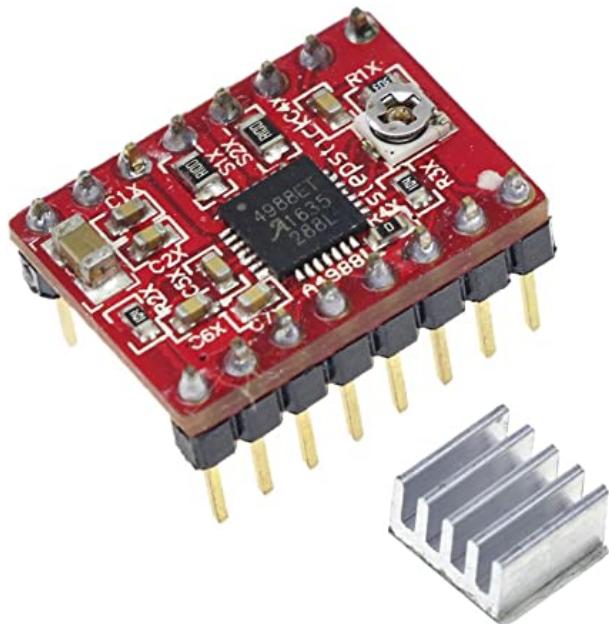


圖. 6.8: A4988 步進馬達驅動器 (A4988 Stepper Motor Driver)



圖. 6.9: 電源 12V/20A(Power supply 12V/20A)

第七章 未來研究建議

本專題以虛實整合為出發點，將實體現有的 3D 列印機系統，透過虛擬環境來架設 3D 列印機，實現模擬現實列印的情況，並透過虛擬場景組裝 uArm 手臂，過程中使用 pySTL 來輔助 CAD 軟體的使用，繪製完的零件可以透過 python 程式達到輕易地更改 stl 檔的位置與大小，無須再打開 CAD 軟體操作，後續研究若是繼續往虛擬環境發展，可以研究 xml 使組合件的物體的位置標準化，往後零件尺寸有更改後無須在虛擬環境再重新組裝，虛擬場景裡就能自動組裝完，透過這項功能，使未來客製化的產品，都能輕易地在虛擬場景完成。

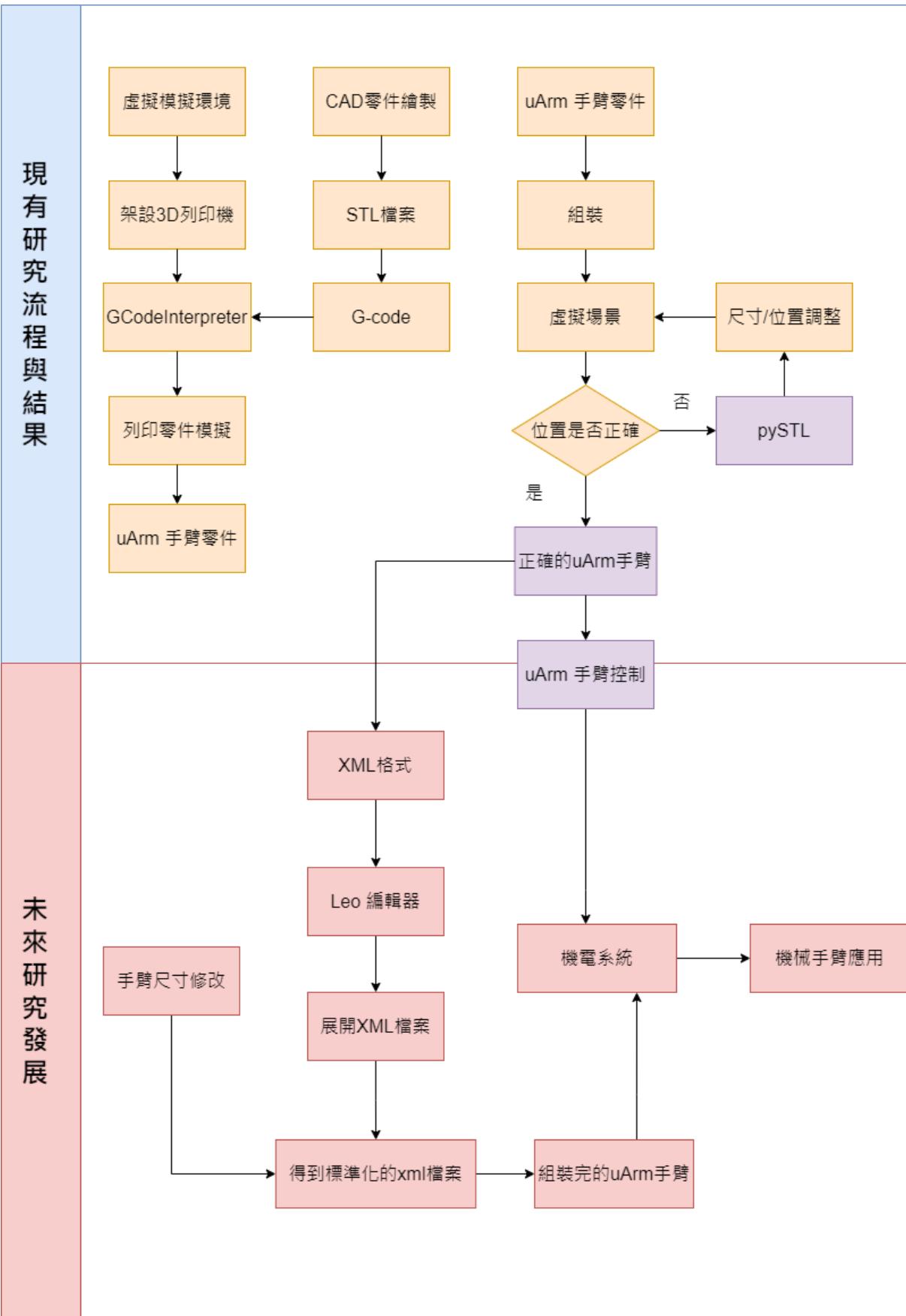


圖. 7.1: 未來研究發展

第八章 問題與討論

Q：我們執行 sample.py 時，python 出現了 print 錯誤。(如：圖 8.1)

```
File "Y:\pySTL.py\sample.py", line 8  
print "Volume " + str(model.get_volume())
```

圖. 8.1: print 錯誤

A：我們必須把 sample.py 與 pySTL.py 裡的全部的 print 加上括號，才能避免上述狀況的發生。(如：圖 8.2)

```
#所有的print都要+() exc  
print ("Volume " + str(model.get_volume()))
```

圖. 8.2: print 加 ()

Q：再次執行 sample.py 時，pySTL.py 出現語法錯誤。(圖.8.3)。

```
File "Y:\pySTL.py\pySTL.py", line 57, in get_centroid  
    if self.centroid == None:
```

圖. 8.3: self.centroid==None 語法錯誤

A：透過網路的搜尋，找到將 == 改成 is，才可以與 None 進行比較，所以我們把 pySTL 裡的 line57 與 line62 的 == 改成 is。(如：圖 8.4)：

	#將兩處地方==改成is,(line57與line62) if self.centroid is None: if self.volume is None:	
--	--	--

圖. 8.4: 將 == 改成 is

Q：再次執行 sample.py 時，pySTL.py 出現語法錯誤。(如：圖 8.3)。

	File "Y:\pySTL.py\pySTL.py", line 196, in write_text_stl f.write("solid {s}\n".format(self.name))	
--	--	--

圖. 8.5: self.name 錯誤

A：仔細讀了 pySTL 裡 line196 上下的代碼，發現到這段的語法是要開啟 ASCII 的 STL 檔案，所以不能使用二進位 (wb) 的方式來寫入檔案，所以把 wb 改成 w，即可排除上述的錯誤。(如：圖 8.4)

	# 將wb改成w f = open(filename, 'w')	
--	-------------------------------------	--

圖. 8.6: 將 w 改成 wb

Q：再次執行 sample.py 時，雖然 python 沒有顯示錯誤，但是寫出的 stl 檔案格式無法開啟。

A：仔細讀了 pySTL 代碼，發現到 line99 是使用 'rb' 來讀取檔案，但是 line104 裡，沒有使用位元 (b) 讀取檔案，導致判斷 ASCII stl 都會變成

binary stl，導致計算 ASCII 尺寸錯誤無法順利開啟檔案，所以改成 b' solid' 即可修正問題。(如：圖 8.5)

	if type==b'solid':	
--	--------------------	--

圖 . 8.7: type==b'solid'

參 考 文 獻

- [1] <https://create.arduino.cc/projecthub/DesiEngineer/how-to-make-a-big-3d-printer-at-home-using-arduino-4a7b79>
- [2] <https://www.coppeliarobotics.com/helpFiles/en/convexDecomposition.htm>
- [3] <https://web.ntnu.edu.tw/~algo/ConvexHull.html>
- [4] <https://github.com/neuebot/FIBR3DEmul>
- [5] [FIBR3DEmulAnOpen-accessSimulat.pdf](#)
- [6] <https://www.ufactory.cc/download-uarm-robot>
- [7] <https://ieeexplore.ieee.org/document/958278>
- [8] <https://github.com/proverbialsunrise/pySTL>
- [9] <https://zh.m.wikipedia.org/zh-tw/STL>
- [10] <https://stackoverflow.com/questions/3289601/referring-to-the-null-object-in-python>
- [11] 896670cd557a9a80de4e568dafcfb6e27b7240.pdf
- [12] <https://hdl.handle.net/11296/48s8f5>
- [13] <https://zh.wikipedia.org/wiki/虛擬化>
- [14] <https://math.stackexchange.com/questions/3932571>
- [15] <https://ieeexplore.ieee.org/document/958278>
- [16] <https://www.researchgate.net/publication/342115737>
- [17] <https://www.cambridge.org/core/journals/proceedings-of-the-design-society-design-conference/article/conception-of-a-digital-twin-in-mechanical-engineering-a-case-study-in-technical-product-development/2E2916B3DC1F42028CD2CDACAB6FA6AC>

附錄

LaTeX

LaTeX 為一種程式語言，支援標準庫 (Standard Libraries) 和外部程式庫 (External Libraries)，不過與一般程式語言不同的是，它可以直接表述 Tex 排版結構，類似於 PHP 之於 HTML 的概念。但是直接撰寫 LaTeX 仍較複雜，因此可以藉由 Markdown 這種輕量的標註式語言先行完成文章，再交由 LaTeX 排版。此專題報告採用編輯軟體為 LaTeX，綜合對比 Word 編輯方法，LaTeX 較為精準正確、更改、製作公式等，以便符合規範、製作。

表. 1: 文字編輯軟體比較表

	相容性	直觀性	文件排版	數學公式	微調細部
LaTeX	√		√	√	√
Word		√			√

- 特點:

1. 相容性：以 Word 為例會有版本差異，使用較高版本編輯的文件可能無法以較低的版本開啟，且不同作業系統也有些許差異；相比 LaTeX 可以利用不同編譯器進行編譯，且為免費軟體也可移植至可攜系統內，可以搭配 Github 協同編譯。
2. 文件排版：許多規範都會要求使用特定版型，使用文字編譯環境較能準確符合規定之版型，且能夠大範圍的自定義排定所需格式，並能不受之後更改而整體格式變形。
3. 數學公式呈現：LaTeX 可以直接利用本身多元的模組套件加入、編輯數學公式，在數學推導過程能夠快速的輸入自己需要的內容即可。
4. 細部調整：在大型論文、報告中有多項文字、圖片、表格，需要調整細部時，要在好幾頁中找尋，而 LaTeX 可以分段章節進行編譯，再進行合併處理大章節。

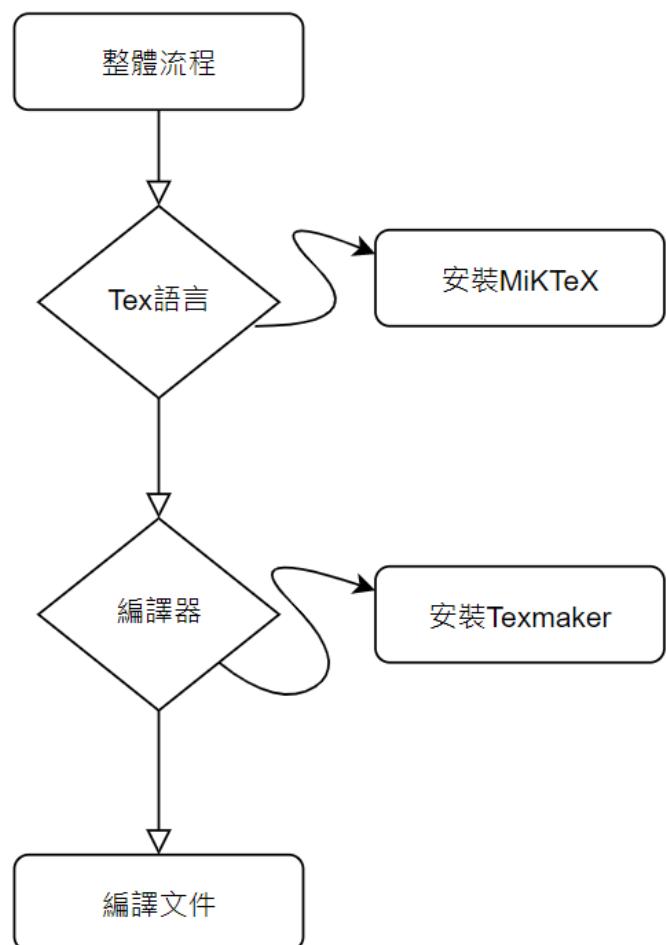


圖. 8: 編譯流程

作者簡介



姓 名：江 東 祐

學 號：40823116

畢業學校：國立虎尾科技大學
機械設計工程系



姓 名：江 建 儒

學 號：40823131

畢業學校：國立虎尾科技大學
機械設計工程系



姓 名：黃 暉 翰

學 號：40823152

畢業學校：國立虎尾科技大學
機械設計工程系



姓 名：蕭 日 傑

學 號：40823153

畢業學校：國立虎尾科技大學
機械設計工程系

【15】

分類編號

:

111-4-APP-3004-1

3D列印模擬在機械手臂設計上的應用

二二一級