

目錄

目錄.....	I
圖目錄.....	II
表目錄.....	III
第一章 程式功能導覽.....	1
第一節 使用者介面與操作步驟.....	1
第二節 程式執行成果展示.....	5
第二章 資料結構介紹.....	11
第一節 點線面清單及關聯性建立.....	11
第二節 Hash map 功能介紹.....	12
第三章 主要函式原理說明.....	17
第一節 尋找自由邊函式.....	17
第二節 尋找模型邊緣函式.....	17
第三節 尋找兩模型面之交線 函式.....	18

圖目錄

圖 1-1 檔案在不同存在狀態下的執行畫面顯示	2
圖 1-2 選擇對讀入的 stl 檔案使用什麼功能之畫面	2
圖 1-3 選擇功能 1 及檔案編號後之畫面	2
圖 1-4 選擇功能 2 及檔案編號後之畫面	3
圖 1-5 選擇功能 3 及檔案編號後之畫面	3
圖 1-6 功能 1 中替選擇的檔案命名之畫面	3
圖 1-7 功能 2 中替選擇的檔案命名之畫面	3
圖 1-8 功能 3 中替選擇的檔案命名之畫面	4
圖 1-9 尋找模型自由邊過程的畫面	4
圖 1-10 尋找模型邊緣過程的畫面	4
圖 1-11 尋找兩模型面之交線過程的畫面	5
圖 1-12 使用 Hash map 前，建立點線面清單及關聯的運算量與運算耗時之關係圖	9
圖 1-13 使用 Hash map 後，建立點線面清單及關聯的運算量與運算耗時之關係圖	9
圖 1-14 尋找模型自由邊的運算量與運算耗時之關係圖	9
圖 1-15 尋找模型邊緣的運算量與運算耗時之關係圖	10
圖 1-16 尋找兩模型面之交線的運算量與運算耗時之關係圖	10
圖 2-1 點線面清單及關聯建立之概念圖	12
圖 2-2 _hash 中重要的成員變數	13
圖 2-3 item 要放置的欄位目前沒有指向任何 item 時的指標操作示意圖	13
圖 2-4 item 要放置的欄位目前有指向 item 時的指標操作示意圖	14
圖 2-5 當要移除的 item 處在欄位裡的中間位置時的指標操作示意圖	15
圖 2-6 當要移除的 item 處在欄位裡的最後位置時的指標操作示意圖	15
圖 2-7 當要移除的 item 處在欄位裡的最前位置時的指標操作示意圖	16
圖 2-8 當要移除的 item 為欄位裡的唯一 item 時的指標操作示意圖	16
圖 3-1 尋找自由邊說明圖	17
圖 3-2 當邊屬於 2 個面時的處理說明圖	18
圖 3-3 當邊不屬於 2 個面時的處理說明圖	18
圖 3-4 兩三角片有可能相交的條件說明圖	19
圖 3-5 兩三角片不可能相交的條件說明圖	19
圖 3-6 當 find_intersect_line 記錄下兩個參數值時的 2 面相交情況	20
圖 3-7 當 find_intersect_line 只記錄下一個參數值時的 2 面相交情況	20
圖 3-8 當 find_intersect_line 沒有記錄下任何參數值時的 2 面相交情況	20
圖 3-9 if_point_in_facet 判斷點是否落於三角片中的方法說明圖	21

表目錄

表 1-1 尋找模型自由邊的成果與運算耗時	5
表 1-2 尋找模型邊緣的成果與運算耗時	6
表 1-3 尋找兩模型面之交線的成果與運算耗時	7

第一章 程式功能導覽

本支程式根據使用者的輸入檔名，讀入指定的 stl 二位元檔，以指標建立三角片的點線面關聯，配合 Hash map 的資料結構，可以快速查找已經存入的點線面資料，避免重複存取。基於建立好的點線面關聯，使用者可以選擇要 尋找模型的自由邊、尋找模型的邊緣 或者 尋找兩模型的面之交線，成果將以 scr 檔案形式輸出，可以在 AutoCAD 中呈現。

第一節 使用者介面與操作步驟

開啟程式執行檔後，使用者首先需輸入要使用的 stl 檔案名稱，若該檔案名稱在程式執行過程已經輸入過，抑或該檔案不存在於執行檔所在的資料夾，畫面都會顯示警示文字作為提醒，如圖 1-1 所示。

當使用者選擇的檔案存在，且檔案名稱為第一次輸入，程式將為該 stl 檔案建立三角片的點線面清單及關聯，完成後會列出清單中的點、線、面數量，如圖 1-1 所示。特此註記，封閉模型的 stl 檔案，面的數量應為線的數量的 $3/2$ 倍，因為每個三角片具有 1 個面、3 個邊、3 個點，而每個邊又為 2 個面所共用；若使用者發現線的數量不為面的 $3/2$ 倍，表示 stl 檔案有毀損情況發生。

```
=====
-----
Enter the input file name.
Input file name (.stl) (default: cylinder.stl): demo
*** Error in opening demo.stl! 檔案不存在的警示文字
Input more file? (y/n) y

=====
-----
Enter the input file name.
Input file name (.stl) (default: cylinder.stl): cube
>>>>>>>> Input file: cube.stl
=====
Establish vertex-edge-facet relationship:
    10 %
    20 %
    30 %
    40 %
    50 %
    60 %
    70 %
    80 %
    90 %
    100 %
    complete!!
12 facets, 18 edgs, 8 vertexes is established!!
Input more file? (y/n) y

=====
-----
Enter the input file name.
Input file name (.stl) (default: cylinder.stl): cube
*** Input file repeatedly! 檔名重複輸入的警示文字
Input more file? (y/n) _
```

圖 1-1 檔案在不同存在狀態下的執行畫面顯示

每處理完 1 份 stl 檔案，程式會詢問使用者是否要再讀入其他 stl 檔案，程式至多能夠替 10 份（開發者可以更改程式提高此上限）存在且不重複的 stl 檔案建立點線面關聯。

當使用者選擇不再讀入其他 stl 檔案或者程式已經替 10 份 stl 檔案建立點線面關聯，程式將會進到下一程序，詢問使用者要對已經讀入的 stl 檔案使用什麼功能，如圖 1-2 所示。

選擇 1，程式可以替模型找到自由邊，並以 scr 檔案形式輸出。然而特此註記，由繪圖軟體輸出的 stl 檔案，屬於封閉模型，在正常情況下並不存在自由邊，因此此功能是針對量測軟體產生的 stl 檔案作處理，也可以用以替毀損的 stl 檔案找尋缺漏三角片的區域。

選擇 2，程式可以替模型找到轉折銳利的邊緣，並以 scr 檔案形式輸出。

選擇 3，程式可以替 2 個模型找到他們的面之交線，並以 scr 檔案形式輸出。

輸入 enter，表示使用默認選項，也就是 2。

```
=====
-----
1) Output free edges of the selected model as script
2) Output edges of the selected model as script
3) Output the intersection lines of the selected models as script
Enter the number of option to select the function to use (default: 2):
>>> Output edges of the selected model as script.
    From input files shown below,
```

圖 1-2 選擇對讀入的 stl 檔案使用什麼功能之畫面

選擇要使用的功能以後，程式接著會列出所有已經建立好點線面關聯的 stl 檔案供使用者選擇，使用者可以輸入要選用的檔案編號，或者輸入 enter 使用默認值。默認值如圖 1-3、圖 1-4、圖 1-5 所示。

```
>>> Output free edges of the selected model as script.
    From input files shown below,
    1) Spherical_Cloud_b.stl,
    2) Spherical noise Cloud_b.stl,
    3) shoe_model.stl,
    4) cube.stl,
    5) cylinder.stl,
    6) big_cylinder.stl,
    7) halfsphere_rl0.stl,
    select the model you want to output free edges (1~7) (default: 1):
>>>>>>>>> Output free edges of Spherical_Cloud_b.stl
```

預設值為第一個檔案

圖 1-3 選擇功能 1 及檔案編號後之畫面

```
>>> Output edges of the selected model as script.
From input files shown below,

1) Spherical_Cloud_b.stl,
2) Spherical noise Cloud_b.stl,
3) shoe_model.stl,
4) cube.stl,
5) cylinder.stl,
6) big_cylinder.stl,
7) halfsphere_r10.stl,
select the model you want to output edges (1~7) (default: 1): 3
>>>>>>>> Output edges of shoe_model.stl
```

預設值為第一個檔案

圖 1-4 選擇功能 2 及檔案編號後之畫面

```
>>> Output the intersection lines of the selected models as script.
From input files shown below,

1) Spherical_Cloud_b.stl,
2) Spherical noise Cloud_b.stl,
3) shoe_model.stl,
4) cube.stl,
5) cylinder.stl,
6) big_cylinder.stl,
7) halfsphere_r10.stl,
select the model you want to output intersection lines (1~7) (default: 1 7): 4 7
>>>>>>>> Output intersection lines between cube.stl and halfsphere_r10.stl
```

預設值為第一個和最後一個檔案

圖 1-5 選擇功能 3 及檔案編號後之畫面

接著，使用者要輸入檔案名稱替即將輸出的 scr 檔案命名，或者輸入 enter 使用默認值，默認值即同於使用者選用的 stl 檔案名稱，如圖 1-6 圖 1-3、圖 1-7、圖 1-8 所示。

```
>>>>>>>> Output free edges of Spherical_Cloud_b.stl
Output file name (.scr) (default: Spherical_Cloud_b.scr): Spherical_Cloud_b_free_edge
>>>>>>>> Output file: Spherical_Cloud_b_free_edge.scr
```

圖 1-6 功能 1 中替選擇的檔案命名之畫面

```
>>>>>>>> Output edges of shoe_model.stl
Output file name (.scr) (default: shoe_model.scr): shoe_model_edge
>>>>>>>> Output file: shoe_model_edge.scr
```

圖 1-7 功能 2 中替選擇的檔案命名之畫面

```

>>>>>>>> Output intersection lines between cube.stl and halvesphere_r10.stl
Output file name (.scr) (default: cube.scr): intersection
>>>>>>>> Output file: intersection.scr

```

圖 1-8 功能 3 中替選擇的檔案命名之畫面

若要使用 尋找模型自由邊 的功能，選擇要使用的 stl 檔案，且輸入完輸出檔名以後，程式會將模型的自由邊以 script 語法輸出成檔案。執行畫面如圖 1-9 所示。

```

>>>>>>>> Output file: Spherical_Cloud_b_free_edge.scr
=====
-----
Find free edges:
    10 %
    20 %
    30 %
    40 %
    50 %
    60 %
    70 %
    80 %
    90 %
   100 %
complete!!
Use more function? (y/n) y

```

圖 1-9 尋找模型自由邊過程的畫面

若要使用 尋找模型邊緣 的功能，選擇要使用的 stl 檔案，且輸入完輸出檔名以後，程式會詢問使用者要找的邊，其所屬的 2 個面的夾角之區間下限及上限為何。輸入 enter 使用默認值，默認值下限為 60 上限為 120。接著，程式會將符合條件的邊以 script 語法輸出成檔案。執行畫面如圖 1-10 所示。

```

>>>>>>>> Output file: shoe_model_edge.scr
=====
-----
Find the target edge.
Enter the angle interval between two related facets (0~180) (default: 60 120):

>>> Output edges that angle between 2 related facets range from 60 to 120 degrees:
    10 %
    20 %
    30 %
    40 %
    50 %
    60 %
    70 %
    80 %
    90 %
   100 %
complete!!
Use more function? (y/n) y

```

圖 1-10 尋找模型邊緣過程的畫面

若要使用 尋找兩模型面之交線 的功能，選擇要使用的 stl 檔案，且輸入完輸出檔名以後，程式會將兩模型的面之交線以 script 語法輸出成檔案。執行畫面如圖 1-11 圖 1-9 所示。

```
>>>>>>>> Output file: intersection.scr
=====
Find intersection lines:
10 %
20 %
30 %
40 %
50 %
60 %
70 %
80 %
90 %
100 %
complete!!
Use more function? (y/n)
```

圖 1-11 尋找兩模型面之交線過程的畫面

每使用完一個功能，程式會詢問使用者要不要繼續使用其他功能，選擇是，將跳回功能選擇單，否，則終止整個程式。

第二節 程式執行成果展示

表 1-1、表 1-2、表 1-3 為幾個範例 stl 檔的執行成果及運算耗時。執行成果皆為 scr 檔案，能夠在 AutoCAD 中呈現；運算耗時的計算是基於運算環境：Windows 10 64 bit、記憶容量 4GB、CPU i5-8250U 1.60GHz、GPU intel UHD Graphics 620。

表 1-1 尋找模型自由邊的成果與運算耗時

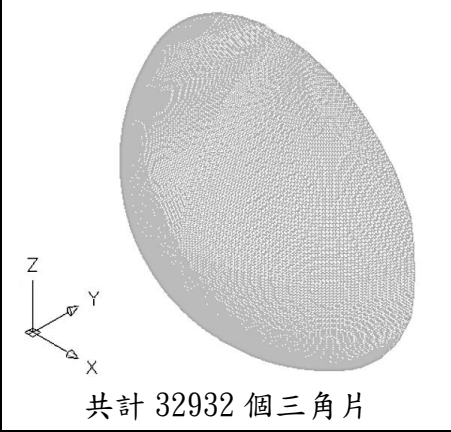
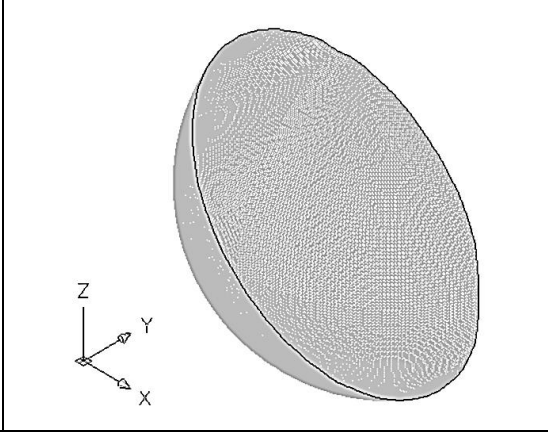
stl 模型	程式尋找到的自由邊	程式運算耗時
		點線面清單及關聯建立： 633ms 尋找自由邊： 9ms

表 1-2 尋找模型邊緣的成果與運算耗時

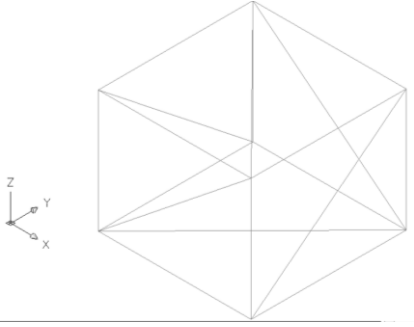
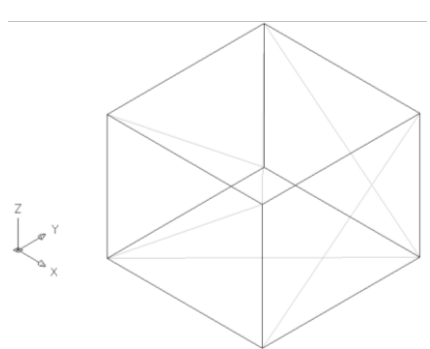
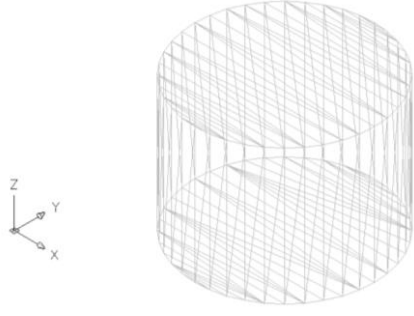
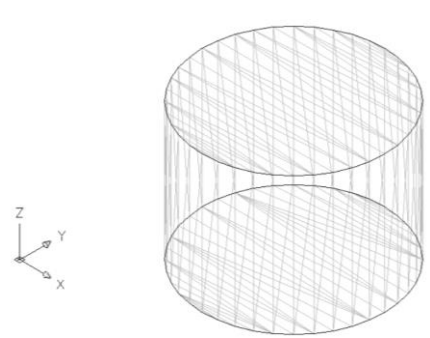
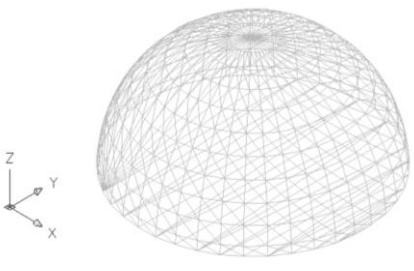
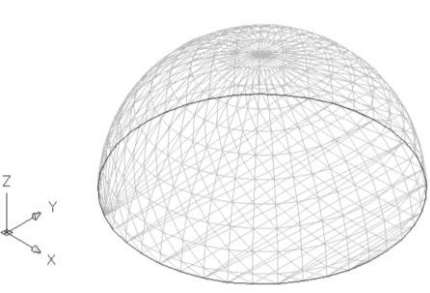
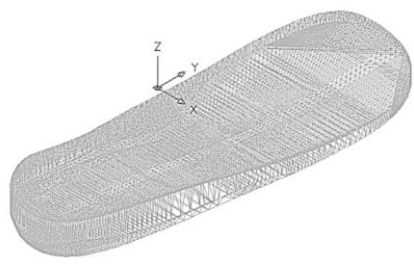
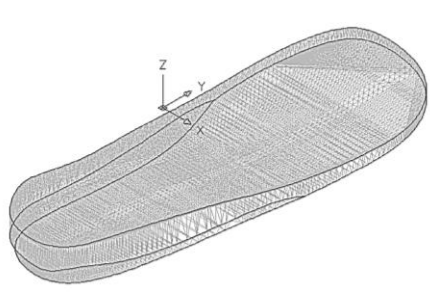
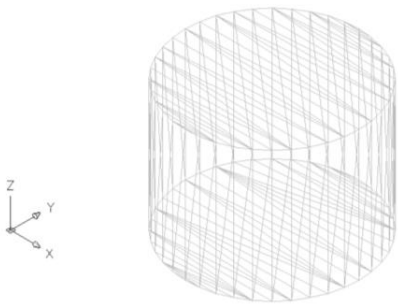
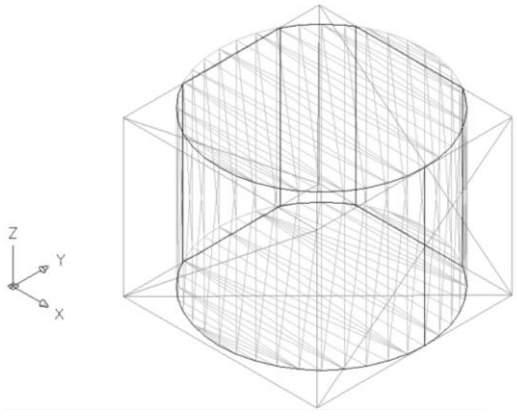
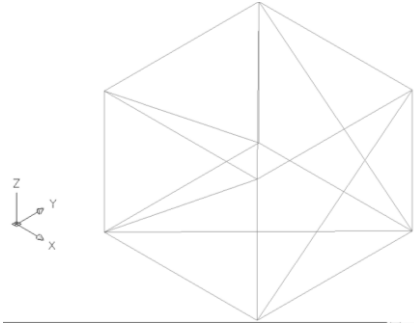
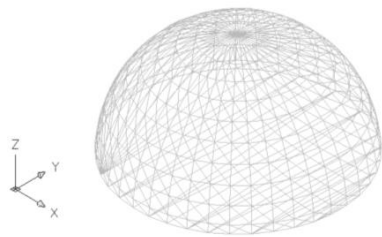
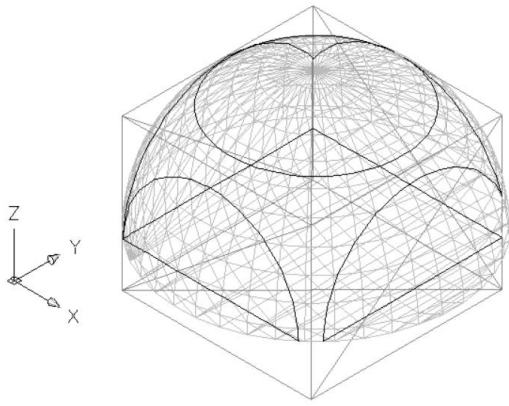
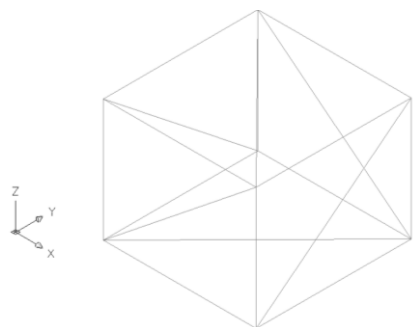
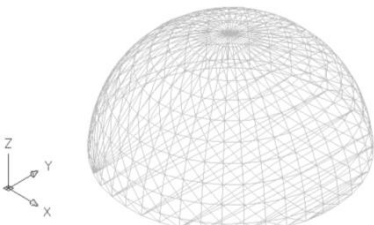
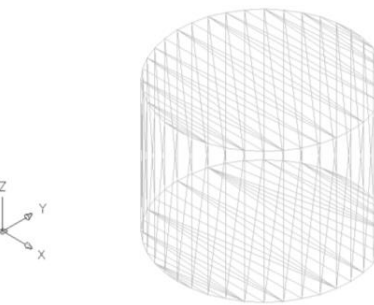
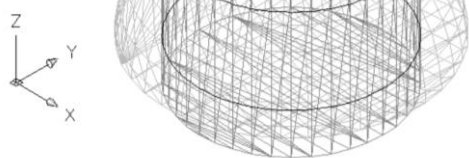
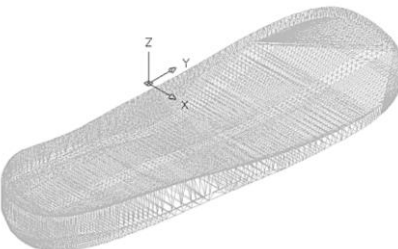
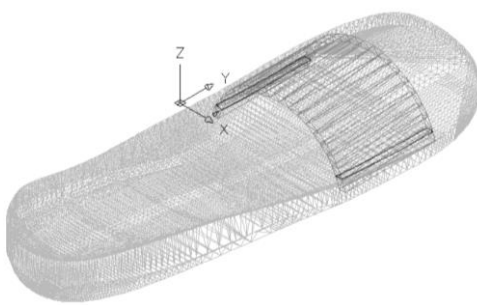
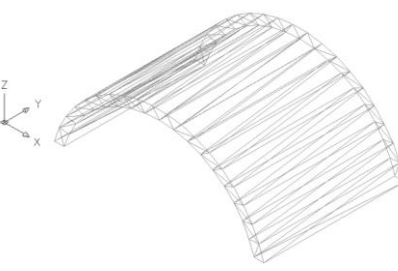
stl 模型	程式尋找到的邊緣	程式運算耗時
 <p>共計 12 個三角片</p>		<p>點線面清單及關聯建立： 16ms</p> <p>尋找邊緣： 4ms</p>
 <p>共計 156 個三角片</p>		<p>點線面清單及關聯建立： 15ms</p> <p>尋找邊緣： 6ms</p>
 <p>共計 900 個三角片</p>		<p>點線面清單及關聯建立： 31ms</p> <p>尋找邊緣： 6ms</p>
 <p>共計 14466 個三角片</p>		<p>點線面清單及關聯建立： 289ms</p> <p>尋找邊緣： 657ms</p>

表 1-3 尋找兩模型面之交線的成果與運算耗時

stl 模型	程式尋找到的兩模型面之交線	程式運算耗時
<p>模型一</p>  <p>共計 156 個三角片</p>		<p>點線面清單及關聯建立：</p> <p>模型一：15ms</p> <p>模型二：16ms</p> <p>尋找兩模型面之交線：</p> <p>11ms</p>
<p>模型二</p>  <p>共計 12 個三角片</p>		
<p>模型一</p>  <p>共計 900 個三角片</p>		<p>點線面清單及關聯建立：</p> <p>模型一：31ms</p> <p>模型二：16ms</p> <p>尋找兩模型面之交線：</p> <p>17ms</p>
<p>模型二</p>  <p>共計 12 個三角片</p>		

<p>模型一</p>  <p>共計 900 個三角片</p>		<p>點線面清單及關聯建立：</p> <p>模型一：31ms</p> <p>模型二：15ms</p> <p>尋找兩模型面之交線：</p> <p>88ms</p>
<p>模型二</p>  <p>共計 156 個三角片</p>		
<p>模型一</p>  <p>共計 14466 個三角片</p>		<p>點線面清單及關聯建立：</p> <p>模型一：289ms</p> <p>模型二：16ms</p> <p>尋找兩模型面之交線：</p> <p>296ms</p>
<p>模型二</p>  <p>共計 152 個三角片</p>		

在建立模型的點線面清單及關聯的過程中，程式會查找新物件是否已存在於物件清單中，避免重複存取物件，這樣的去重功能，減少了約 80% 的點物件數量及約 50% 的線物件數量。

然而若存放物件進陣列時都不做分類，每次查找物件的時間複雜度都會是 $O(n^2)$ ；使用 Hash map 將物件做分類，能夠使查找物件的時間複雜度降至 $O(n)$ 。圖 1-12、圖 1-13 分別為使用 Hash map 前後，建立點線面清單及關聯的運算量與運算耗時之關係圖。

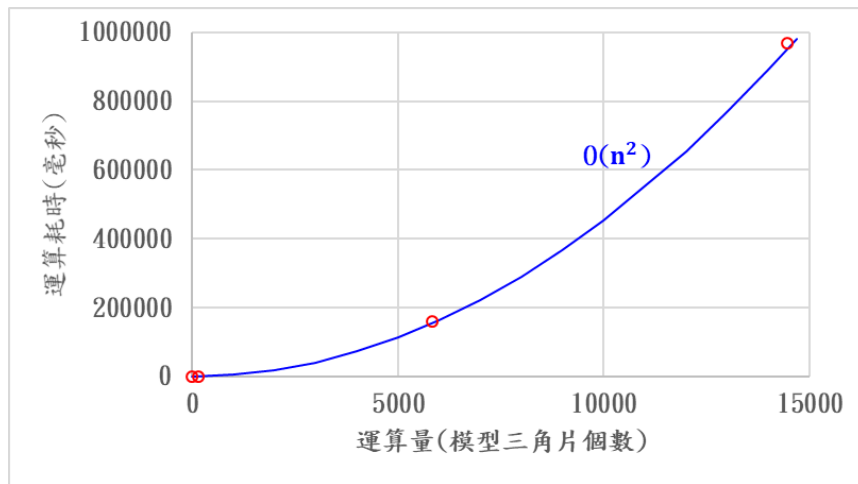


圖 1-12 使用 Hash map 前，建立點線面清單及關聯的運算量與運算耗時之關係圖

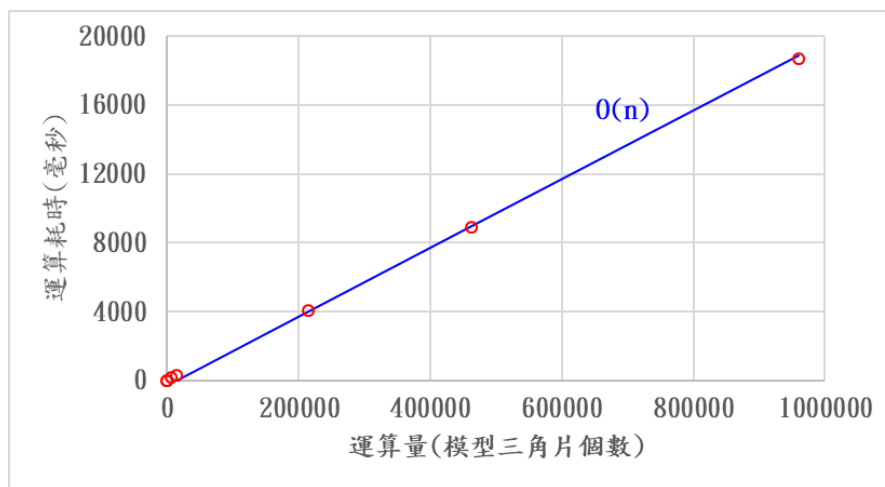


圖 1-13 使用 Hash map 後，建立點線面清單及關聯的運算量與運算耗時之關係圖

另外圖 1-14、圖 1-15、圖 1-16 也列出本程式的三項主要功能所需的運算時間與時間複雜度。

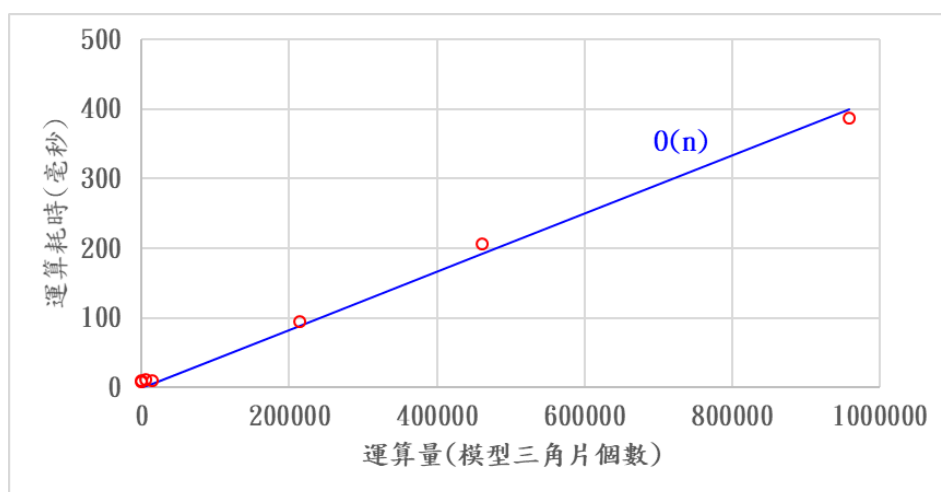


圖 1-14 尋找模型自由邊的運算量與運算耗時之關係圖

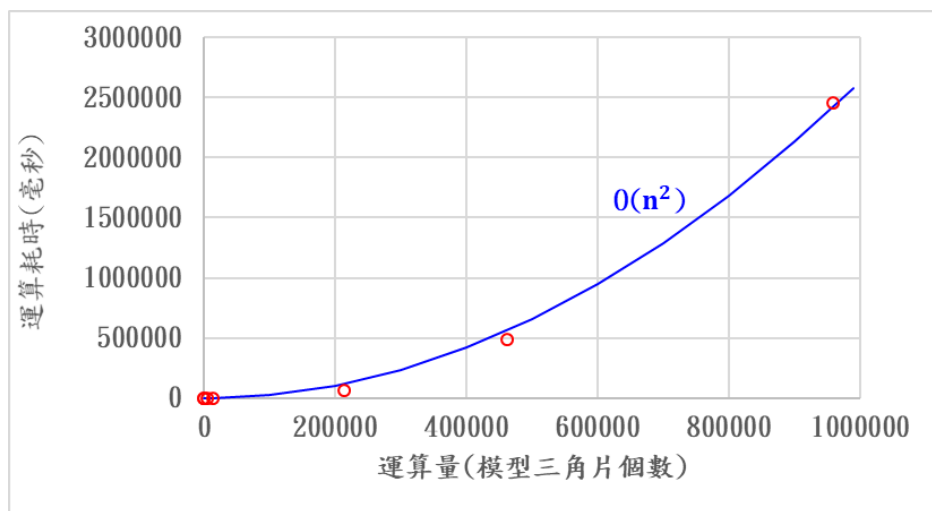


圖 1-15 尋找模型邊緣的運算量與運算耗時之關係圖

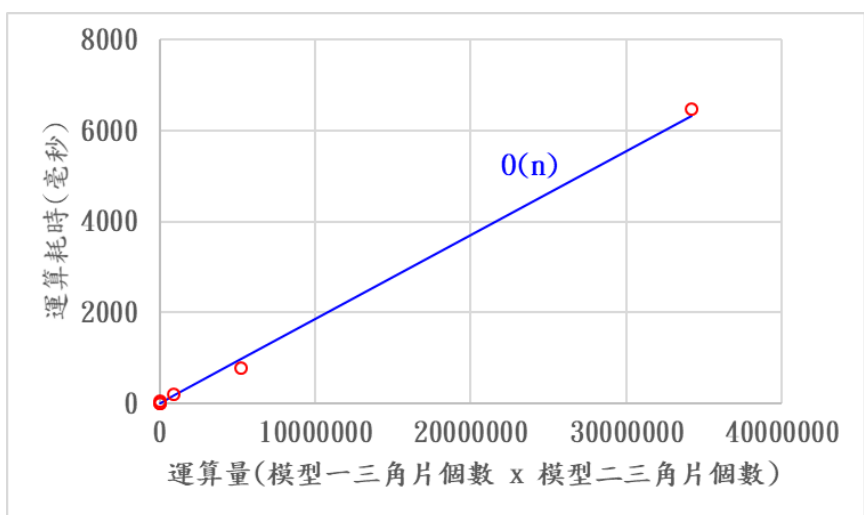


圖 1-16 尋找兩模型面之交線的運算量與運算耗時之關係圖

第二章 資料結構介紹

stl 檔案透過三角網格外包實體或曲面表面以近似出模型外觀，其檔案主要內容包含總三角網格數、每個三角片的面法向量及 3 個頂點座標。然而相鄰的三角片因為共用邊線與頂點，導致邊線與頂點的資料重複存取，浪費儲存空間也不便於查找。因此本支程式在將 stl 檔資料讀入後，將點、線、面 分別存入 點、線、面 的 Hash map 建立清單，透過 Hash map 的資料結構，可以快速查找 點、線、面 是否已經存在於清單中，避免資料重複存取。

另外，程式透過指標建立點線面彼此的關聯，即記錄每個點屬於哪些線和面、每個線屬於哪些點和面、每個面屬於哪些線和點，方便後續的應用。

第一節 點線面清單及關聯性建立

在本支程式中，點線面的清單及關聯性是透過 `read_stl_and_establish_vertex_edge_facet_relationship` 函式所建立，其執行概念如圖 2-1 所示。配合 stl 的檔案格式，程式每次會讀入一個三角片的資訊，整理後加以存取。

每讀入一個三角片的資訊，程式會產生 3 個點物件指標指向點物件、3 個邊物件指標指向邊物件、1 個面物件指標指向面物件。點物件由讀入的三角片之 3 個頂點座標建構，邊物件由 2 個點物件建構，面物件由 3 個點物件建構。指標所指向的物件建構完成後，在物件清單中以成員函式 `FindItem` 查找物件是否存在，若不存在則以成員函式 `AddItem` 加入物件，若物件已經存在於清單中，則以清單中的物件取代新建構的物件。

將三角片的點線面放入清單後，我們透過 點、線、面 物件中的成員變數來建立點線面的關聯。點物件的成員變數中，`facet_of_vertex` 用於存取點所屬於的面物件，`edge_of_vertex` 用於存取點所屬於的邊物件；邊物件的成員變數中，`vertex_of_edge` 用於存取邊所屬於的點物件，`facet_of_edge` 用於存取邊物件所屬於的面物件；面物件的成員變數中，`vertex_of_facet` 用於存取面所屬於的點物件，`edge_of_facet` 用於存取邊物件所屬於的面物件。

事先查找物件是否已存在於清單中再決定要否加入新物件，不但減少重複物件存取所浪費的空間，也讓點線面的關聯得以統整。

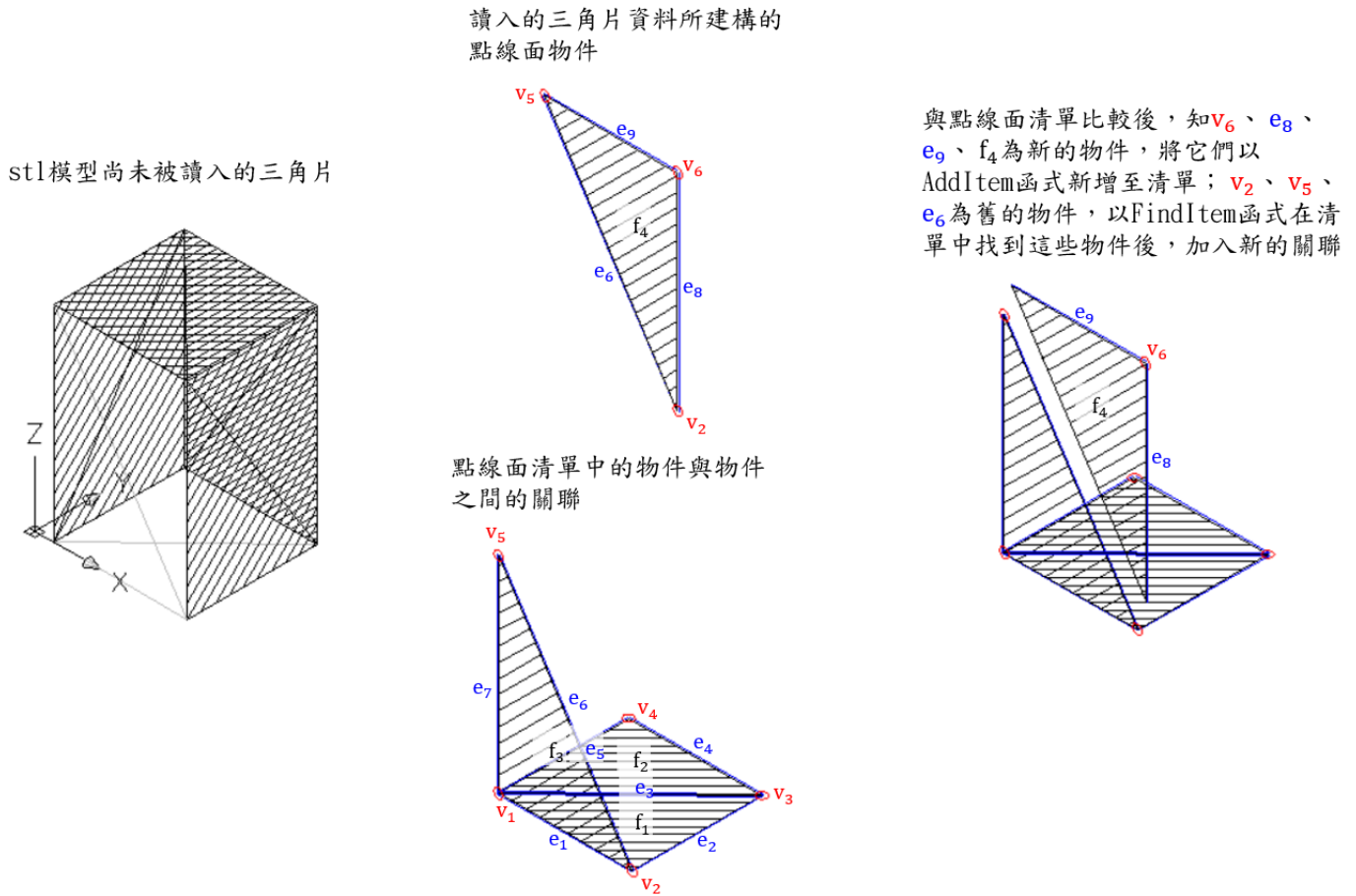


圖 2-1 點線面清單及關聯建立之概念圖

第二節 Hash map 功能介紹

Hash map 是一種方便大量資料快速查找的資料結構，在將物件放入 Hash map 之前，會先經由一簡單且運算快速的 Hash 函式處理物件，決定物件要放在 Hash map 陣列的第幾欄位中。陣列的每一欄位能夠存放多個物件，並透過指標將這些物件串連在一起，以共後續查找。特此註記，一個好的 Hash map 所具備的 Hash 函式，能夠使物件均勻放置在各欄位，讓每欄位中的物件數量相近。

本程式使用 Hash map 製作點、線、面 的清單，分別存放點、線、面物件。對於點、線、面 清單，由於加入物件、找尋物件、刪除物件的動作都十分雷同，所以本支程式首先寫了一個模板類別 _hash，再由此類別衍生出類別 vertex_hash、edge_hash、facet_hash。衍生出的類別繼承了類別 _hash 中的建構式、成員變數、成員函式 AddItem、FindItem、RemoveItem，並且發展各自的 Hash 函式以對不同類別物件做處理。

類別 _hash 中重要的成員變數如圖 2-2 所示。其中包含 item，用來存放物件及指向前後 item 的指標；HashTable，為存放 item 的陣列；tableSize，定義了 HashTable 陣列的大小。使用者建構物件清單時若使用無引數的建構式，則 tableSize 預設為 181；或者可以使用有引數的建構式給定 tableSize 值。

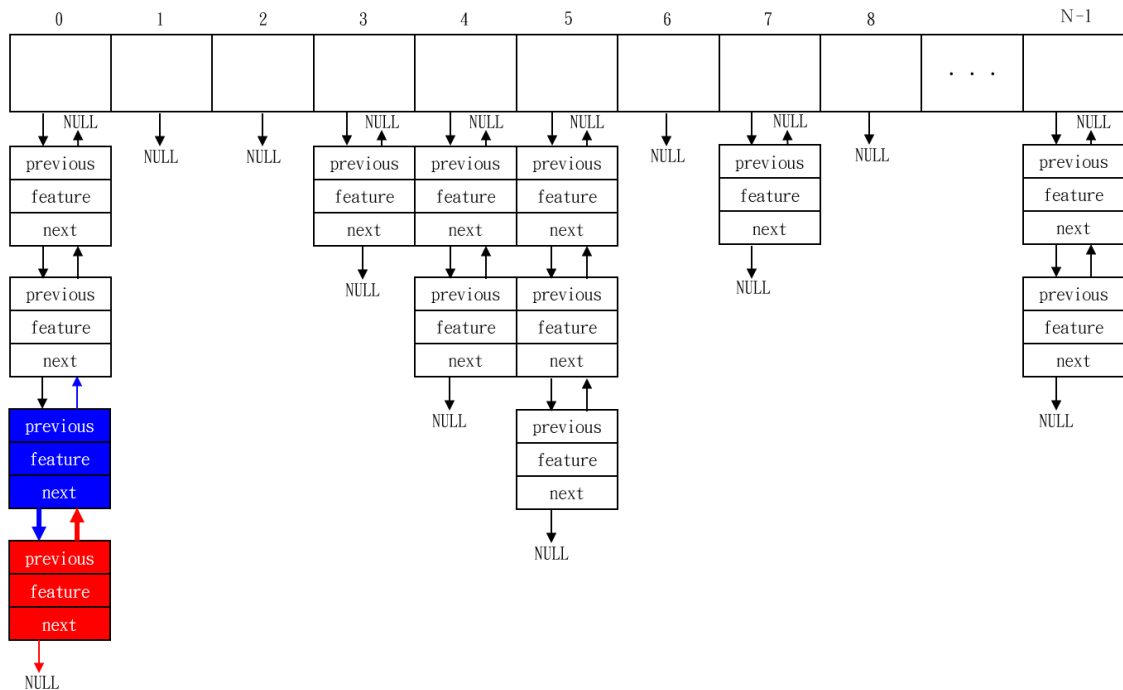


圖 2-4 item 要放置的欄位目前有指向 item 時的指標操作示意圖

當使用者欲使用 FindItem 函式尋找物件時，FindItem 首先會呼叫 Hash 函式處理物件，了解物件如果存在，應該放置在 HashTable 的哪個欄位。若遍歷了該欄位中的所有 item 仍未找到物件，則回傳 NULL；否則回傳找到的物件。

RemoveItem 的實作原理如圖 2-5、圖 2-6、圖 2-7、圖 2-8 所示，其中紅色部分為要移除的 item，藍色部分為其他要做出相應變動的 item，粗體的箭頭表示需要更動的指標。當使用者欲使用 RemoveItem 函式移除物件時，RemoveItem 首先會呼叫 Hash 函式處理物件，了解物件如果存在，應該放置在 HashTable 的哪個欄位。若遍歷了該欄位中的所有 item 仍未找到物件，表示要移除的物件其實不存在，則不做任何動作；若物件存在，則將針對各種不同情況做相應處理，情況及相應的處理方法條列如下：

- 1) 若物件所屬的 item 的 previous 和 next 都有指向其他 item，表示要移除的 item 在欄位的所有 item 中，是處在一中間的位置。此時，我們要將要移除的 item 的前一個 item 的 next 指向要移除的 item 的下一個 item；另將要移除的 item 的下一個 item 的 previous 指向要移除的 item 的前一個 item；最後再將要移除的 item 刪去。

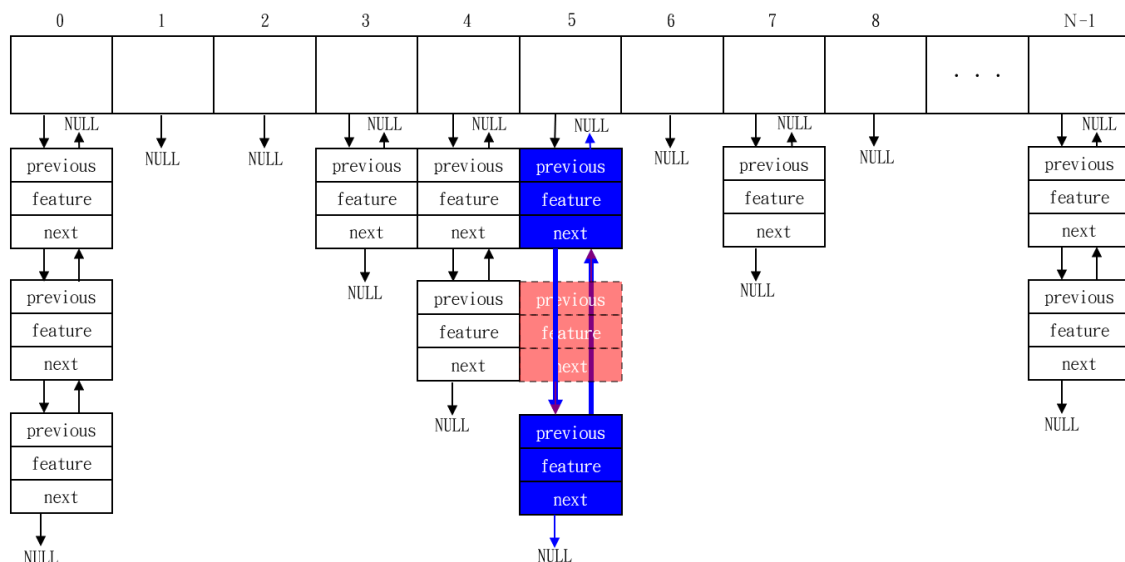


圖 2-5 當要移除的 item 處在欄位裡的中間位置時的指標操作示意圖

- 2) 若物件所屬的 item 的 previous 有指向其他 item，但 next 沒有指向其他 item，表示要移除的 item 在欄位的所有 item 中，是處在一最後的位置。此時，我們需要將要移除的 item 的前一個 item 的 next 指向 NULL；最後再將要移除的 item 刪去。

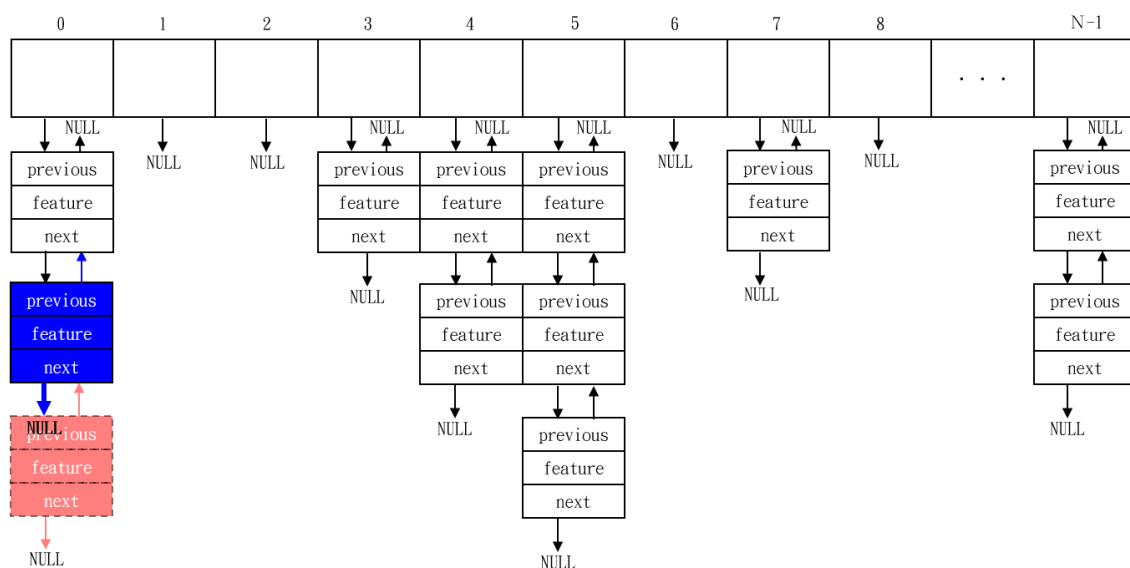


圖 2-6 當要移除的 item 處在欄位裡的最後位置時的指標操作示意圖

- 3) 若物件所屬的 item 的 previous 沒有指向其他 item，但 next 有指向其他 item，表示要移除的 item 在欄位的所有 item 中，是處在一最前面的位置。此時，我們需要將要欄位指向要移除的 item 的下一個 item；另將要移除的 item 的下一個 item 的 previous 指向 NULL；最後再將要移除的 item 刪去。

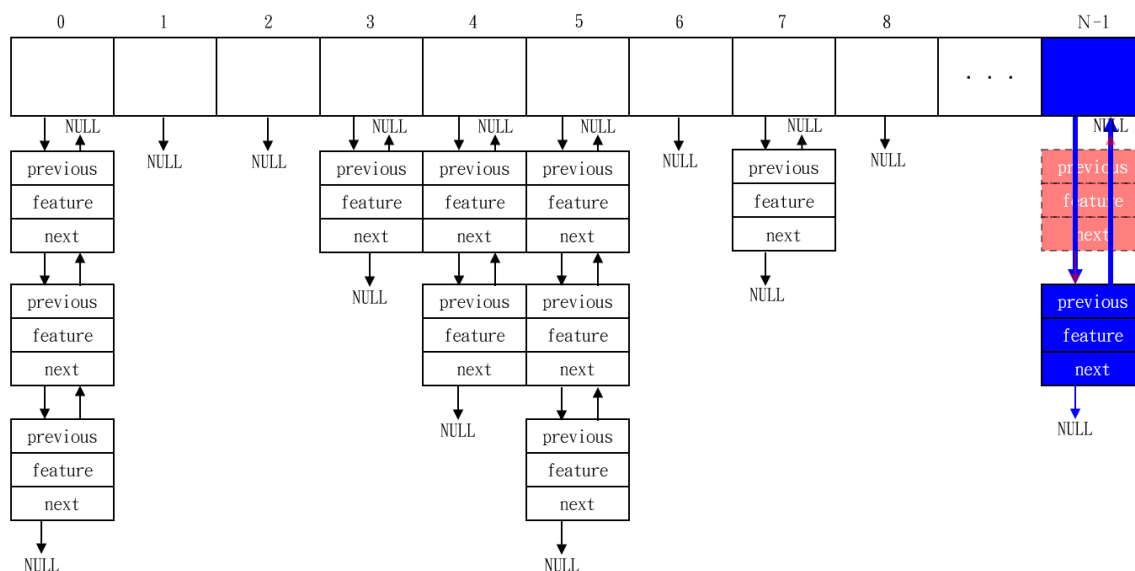


圖 2-7 當要移除的 item 處在欄位裡的最前位置時的指標操作示意圖

- 4) 若物件所屬的 item 的 previous 和 next 都沒有指向其他 item，表示要移除的 item 是欄位中的唯一 item。此時，我們將欄位指向 NULL；最後再將要移除的 item 刪去。

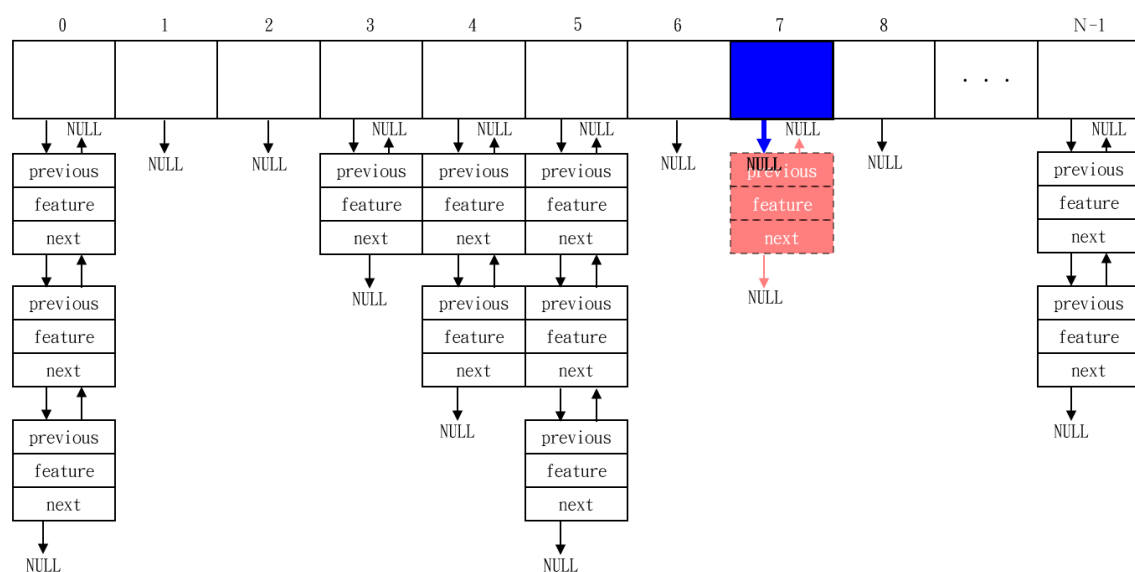


圖 2-8 當要移除的 item 為欄位裡的唯一 item 時的指標操作示意圖

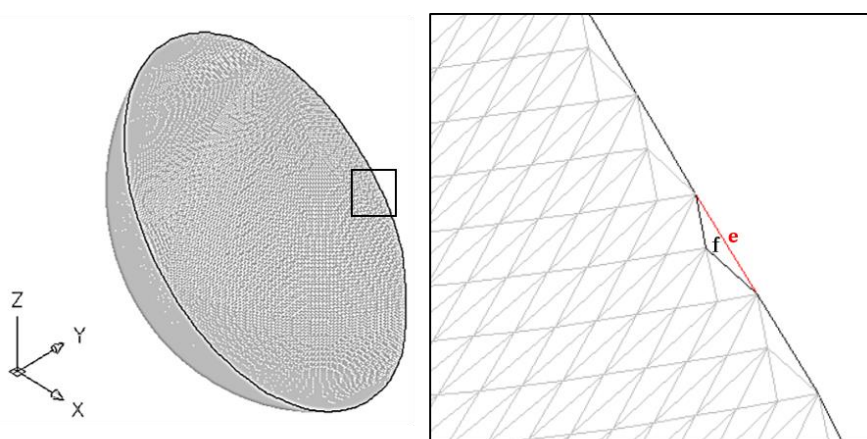
另外，如先前所提到的，vertex_hash、edge_hash、facet_hash 雖然都繼承自類別_hash，但擁有不同的 Hash 函式，而他們的 Hash 函式都是透過將物件本身的點座標處理後，除以 tableSize 得到的餘數，作為物件要放在哪個欄位的依據。物件本身的點座標處理方式為，將點的 x、y、z 座標相加後乘以 10^6 ，這樣確保點座標在處理過後的數值大於 tableSize，避免了多數欄位閒置的情況發生；另外之所以乘以 10^6 是因為 stl 檔案中的點座標都是以 float 的資料型態儲存，故只有 6 位有效數字，而利用小數點後數字的雜亂分布，可以讓物件得以均勻放置於 HashTable 之中。根據測試，HashTable 每個欄位最多不會放置超過 10 個物件，最少為 0 個物件。

第三章 主要函式原理說明

針對本支程式提供給使用者使用的三個功能：尋找自由邊、尋找模型邊緣、尋找兩模型面之交線，本章節將介紹這些功能的實施細節。

第一節 尋找自由邊函式

尋找自由邊的功能由類別 `edge_hash` 中的成員函式 `draw_free_edge` 達成，其實施原理圖 3-1 所示。在 `draw_free_edge` 中，`HashTable` 中所有 item 被遍歷，並查看每一 item 中的 `feature` (在此，類別為 `edge`)，判斷邊是否只屬於 1 個面 (`facet_of_edge.size() == 1`)，是則表示此邊為一自由邊，並以 `script` 的語法輸出該邊；否則掠過該邊。



e只屬於面**f**一個面，故**e**是一自由邊

圖 3-1 尋找自由邊說明圖

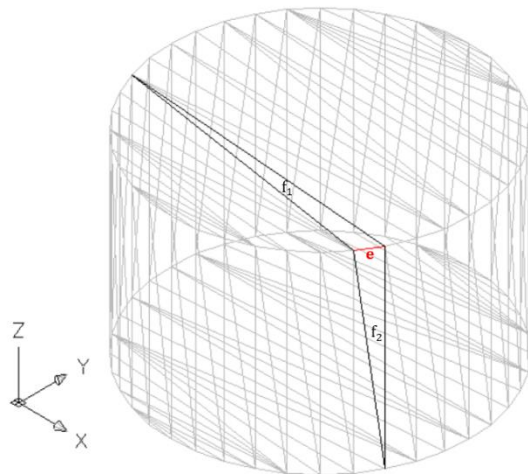
第二節 尋找模型邊緣函式

尋找模型邊緣的功能由類別 `edge_hash` 的朋友類別 `sort_edge_for_edge_hash` 達成。`sort_edge_for_edge_hash` 除了是 `edge_hash` 的朋友，也繼承自類別 `_hash`，因此它除了有自己的 `HashTable`，也可以使用 `edge_hash` 的 `HashTable`。

`sort_edge_for_edge_hash` 除了製作自己的成員函式 `Hash`，另有兩個成員函式 `sort_edge` 及 `draw_edge`。`sort_edge` 所做的事，其實就是將 `edge_hash` 中的所有物件，透過 `AddItem` 放入自己的 `HashTable`。然而 `sort_edge_for_edge_hash` 與 `edge_hash` 有十分不同的 `Hash` 函式，因此物件在這樣的移動過程被排序，產生截然不同的 `HashTable` 內容。

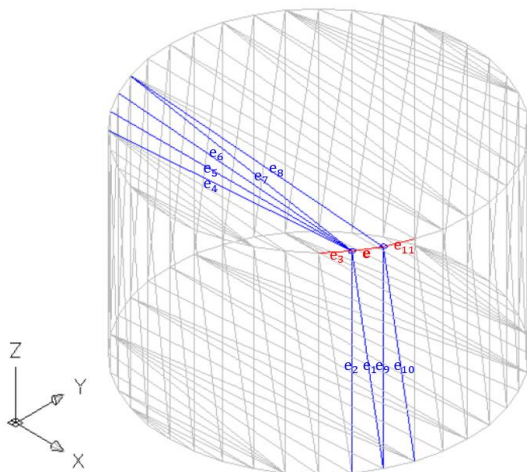
在 `sort_edge_for_edge_hash` 的 `Hash` 函式中，判斷邊物件要放在 `HashTable` 中的哪個欄位的依據在於，邊所在位置的 2 個面夾角幾度。然而不是每個邊都會屬於 2 個面，在 `stl` 檔毀損的情況下，有些邊只會屬於 1 個面。因此，如圖 3-2、圖 3-3 所示，函式首先判別邊物件是否屬於 2 個面，若邊屬於 2 個面，則直接計算這 2 個面的法向量夾角並回傳，意即面法向量夾角的度數即邊要放在的 `HashTable` 欄位；若邊不屬於 2 個面，則透過邊上的兩個點，找到屬於這 2 個點的其他邊，即找尋邊的其他所有相鄰邊，只要相鄰邊與邊的夾角夠小，我們則當作相鄰邊為邊的延伸，那麼邊與相鄰邊理應有一樣的面夾角，因此回傳相鄰邊的面法向量夾角；若邊不屬於 2 個面，且邊的所有相鄰邊，不是與邊的夾角都不夠小而不能視為邊的延伸，就是不屬於兩個面而計算不出面法向量夾

角，則回傳 180，將此一有誤項放在 HashTable 最後一欄中。特此註記，sort_edge_for_edge_hash 只繼承_hash 的無引數的建構式，因此 tableSize 給定為 181，此 181 就是特地設計用以存放面法向量夾角 0~180 度的邊。



當邊 e 屬於2面 f_1 、 f_2 ，計算 f_1 、 f_2 法向量的夾角度數，此度數即 e 要放在 HashTable 中的欄位

圖 3-2 當邊屬於 2 個面時的處理說明圖



當邊 e 不屬於2面，透過 e 的2端點找到 e 的其他相鄰邊 ($e_1 \sim e_{11}$)，其中只有 e_3 、 e_{11} 與 e 的夾角夠小，能夠當作 e 的延伸。若 e_3 、 e_{11} 屬於2面，計算其面法向量夾角， e 的面法向量夾角理應等同於此

圖 3-3 當邊不屬於 2 個面時的處理說明圖

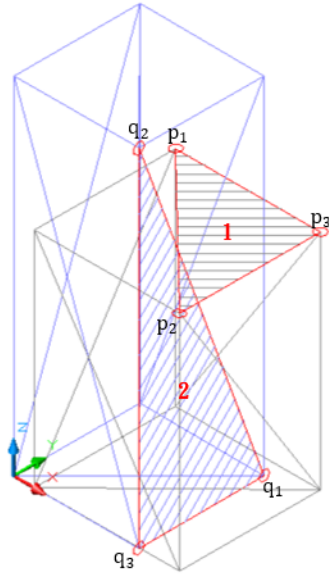
在邊排序好之後，程式會把使用者輸入的面夾角轉換為面法向量夾角，透過成員函式 draw_edge 以 script 的語法輸出指定夾角區間的邊。

第三節 尋找兩模型面之交線 函式

尋找兩模型面之交線的功能由類別 facet_hash 中的成員函式 draw_intersection_line 達成。draw_intersection_line 會遍歷該 facet_hash 的 HashTable 中所有 item，對於每一 item 又遍歷另一份 facet_hash 的 HashTable 中所有 item，以此來比對兩份 facet_hash 中 item 的面物件。若 2 個來自不同 facet_hash 的面物件透過 if_facets_are_possible_to_intersect 運算後，發現彼此有可能相交，將進一步透過 find_intersect_line 找出兩面的交線，若兩面確實有相交，將回傳交線兩端點的座標，並以 script 的語法輸出交線線段；若兩面其實沒有相交，將回傳 NULL，則不採取任何動作。特此註記，if_facets_are_possible_to_intersect 及 find_intersect_line 都是在

做面與面間的運算，是故兩者都屬於類別 facet 中的成員函式。

if_facets_are_possible_to_intersect 的實施原理如圖 3-4、圖 3-5 所示。對於 2 個三角片，我們都可以列出他們各自的平面方程式，也知道他們的頂點座標。今天若 2 個三角片的其中 2 個頂點分別在另一三角片構成的平面兩端，則表示三角片有相交的機會。此判斷函式運算快速，有效篩去不可能相交的兩面物件，避免程式浪費時間運算所有面物件之交線，大幅提升運算速度。



三角片2的三頂點(q_1, q_2, q_3)與三角片1的平面($n_{1x}x + n_{1y}y + n_{1z}z = C_1$)之關係：

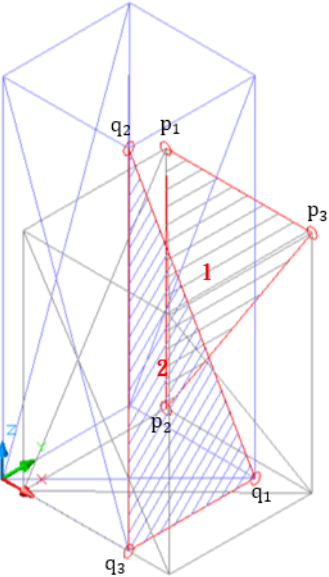
$$\begin{aligned} n_{1x}q_{1x} + n_{1y}q_{1y} + n_{1z}q_{1z} - C_1 &< 0 \\ n_{1x}q_{2x} + n_{1y}q_{2y} + n_{1z}q_{2z} - C_1 &> 0 \\ n_{1x}q_{3x} + n_{1y}q_{3y} + n_{1z}q_{3z} - C_1 &< 0 \end{aligned}$$

三角片1的三頂點(p_1, p_2, p_3)與三角片2的平面($n_{2x}x + n_{2y}y + n_{2z}z = C_2$)之關係：

$$\begin{aligned} n_{2x}p_{1x} + n_{2y}p_{1y} + n_{2z}p_{1z} - C_2 &< 0 \\ n_{2x}p_{2x} + n_{2y}p_{2y} + n_{2z}p_{2z} - C_2 &> 0 \\ n_{2x}p_{3x} + n_{2y}p_{3y} + n_{2z}p_{3z} - C_2 &> 0 \end{aligned}$$

上三式之中，出現不同正負號，且下三式之中，也出現不同正負號，表示兩三角片有相交之可能

圖 3-4 兩三角片有可能相交的條件說明圖



三角片2的三頂點(q_1, q_2, q_3)與三角片1的平面($n_{1x}x + n_{1y}y + n_{1z}z = C_1$)之關係：

$$\begin{aligned} n_{1x}q_{1x} + n_{1y}q_{1y} + n_{1z}q_{1z} - C_1 &< 0 \\ n_{1x}q_{2x} + n_{1y}q_{2y} + n_{1z}q_{2z} - C_1 &< 0 \\ n_{1x}q_{3x} + n_{1y}q_{3y} + n_{1z}q_{3z} - C_1 &< 0 \end{aligned}$$

三角片1的三頂點(p_1, p_2, p_3)與三角片2的平面($n_{2x}x + n_{2y}y + n_{2z}z = C_2$)之關係：

$$\begin{aligned} n_{2x}p_{1x} + n_{2y}p_{1y} + n_{2z}p_{1z} - C_2 &< 0 \\ n_{2x}p_{2x} + n_{2y}p_{2y} + n_{2z}p_{2z} - C_2 &< 0 \\ n_{2x}p_{3x} + n_{2y}p_{3y} + n_{2z}p_{3z} - C_2 &> 0 \end{aligned}$$

上三式具有相同正負號，所以不論下三式的正負號有否一致，兩三角片都無相交之可能

圖 3-5 兩三角片不可能相交的條件說明圖

find_intersect_line 首先透過 plane_intersection 計算 2 三角片的平面方程式之相交線參數式，接者將 2 三角片的共 6 條邊也表示為參數式，透過 line_intersection 計算這 6 條邊與方才的交線之交點。只有當交點落在 2 三角片中(if_point_in_facet() == 1)，我們會記錄下相交線之參數值。運算完 6 條邊與相交線之交點及參數後，我們取最大範圍的參數區間，並將區間極值帶入相

交線的參數式，算出線段兩端點的座標加以回傳(如圖 3-6 之情況)；如果記錄下的參數值只有一個(如圖 3-7 之情況)，抑或沒有任何參數被記錄下來(如圖 3-8 之情況)，表示並沒有成功找到三角片的相交線段，回傳 NULL。

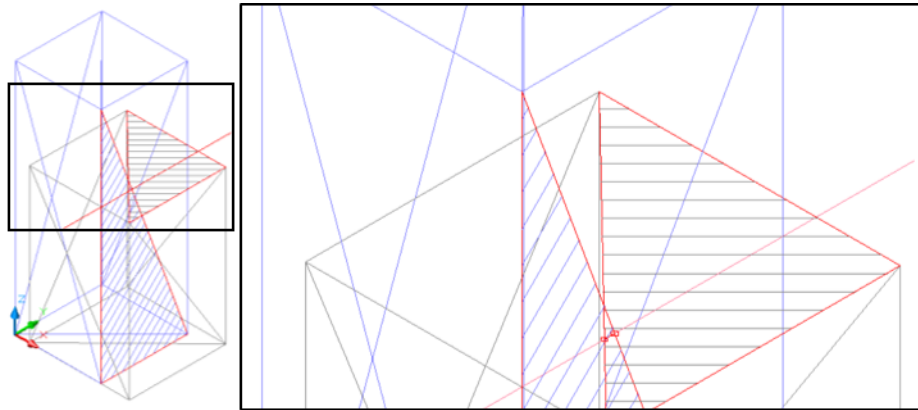


圖 3-6 當 find_intersect_line 記錄下兩個參數值時的 2 面相交情況

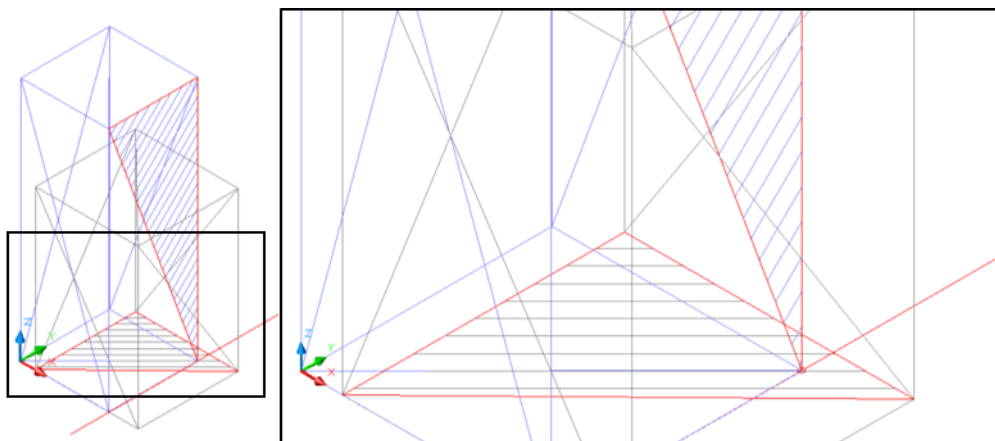


圖 3-7 當 find_intersect_line 只記錄下一個參數值時的 2 面相交情況

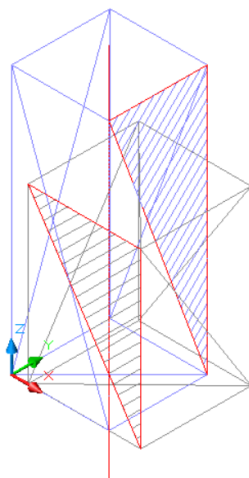


圖 3-8 當 find_intersect_line 沒有記錄下任何參數值時的 2 面相交情況

if_point_in_facet 是類別 facet 中，判斷點是否落在該面中的成員函式，其實施原理如圖 3-9 所示。當點落在三角片形成的面中（包含落在邊線上），點與三角片之 3 個頂點的連線可以將三角片分割成 3 個區塊，這 3 個區塊的面積加總，會等於透過三角片的 2 個邊內積出的三角片面積。如果兩種計算三角片面積的方式，算出的面積值不同，表示該點不落在三角片中。

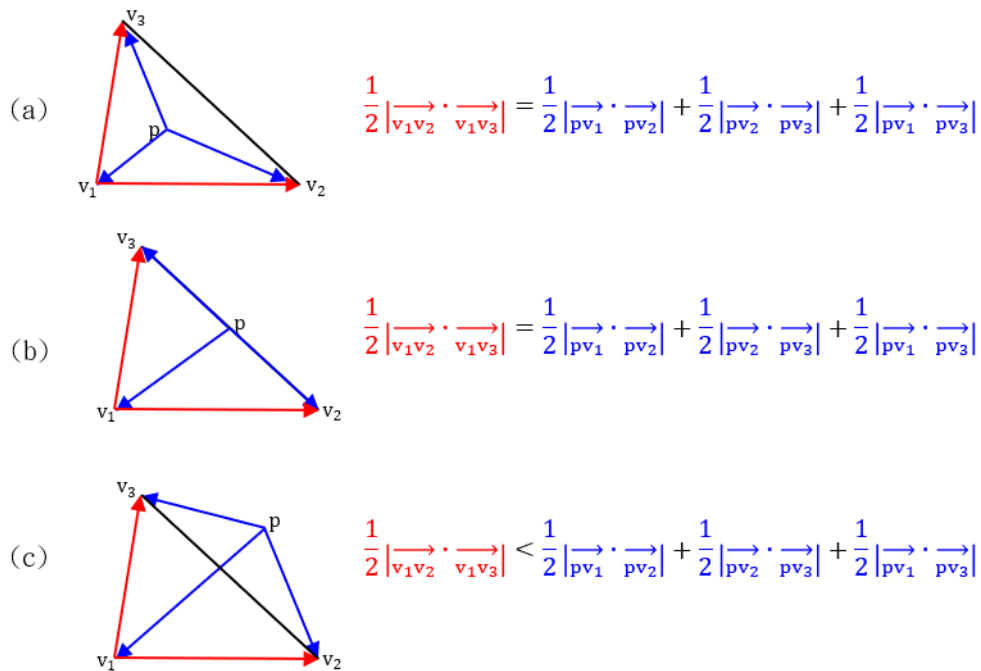


圖 3-9 if_point_in_facet 判斷點是否落於三角片中的方法說明圖

(a)點落於三角片中的條件式(b)點落於三角片邊線上的條件式(c)點落於三角片外的條件式

plane_intersection 函式能夠計算 2 三角片的平面方程式之相交線參數式。只要 2 平面不平行，透過 2 面法向量的外積，能夠計算出 2 平面交線的向量；交線上的點則可透過聯立 2 平面方程求解。然 2 方程式無法求解 3 未知(x, y, z)，因此擇 1 未知設為 0 帶入 2 方程式，簡化後的 2 方程式若為線性獨立，則可以使用克拉馬公式 CramersFormula 得到另 2 未知的唯一解。

line_intersection 函式能夠計算 2 線參數式的交點及相交情況下的參數。2 線要有交點，則 2 線參數式的 x、y、z 項都需相等，亦即可以列出 3 個等式，然而實則只有 2 個參數要求解，選擇 2 線性獨立的關係式，則可以使用克拉馬公式 CramersFormula 得到 2 線相交時的 2 線參數式的參數之唯一解。

最後，提醒開發者，任何一數值有曾經經歷過運算，精度就有可能遺失，因此在程式開發過程中，遇到要比對兩數值是否相等的情形時，需要以兩數值相差是否小於一極小值($|x_1 - x_2| < \epsilon$)，取代兩數值是否相等($x_1 == x_2$)，而本程式中設定的極小值為 10^{-5} 或 10^{-6} ，此為配合 float 資料型態之精度及考慮運算所需的嚴謹性所定。