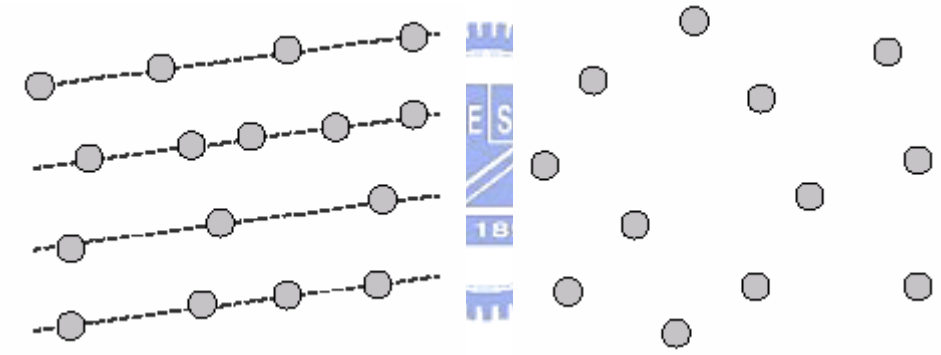
第二章 三維資料結構及檔案格式

2-1 三維資料結構

三維資料之定義為空間中一群具有三維座標（x, y, z）之點位，透過三維座標之描述，可得知點位在空間中分佈之情形，但此時的點資料為不規則分佈，其中也包含許多不必要的雜訊，須先經過資料濾除及資料結構化的程序後，方能從不規則的點雲資料中萃取出使用者所需的資訊，進而描述空間中之三維物體。

在逆向工程技術建構模型過程中，所獲得的資料均是以點雲的型態呈現，依掃描方式不同，可分為掃描線點資料，即依掃描線方向依序記錄點資料（如圖 2-1（a））及無順序點資料（如圖 2-1（b））（蔡政霖，2003）。



（a）掃描線點資料　　（b）無順序點資料

圖 2- 1 掃描點雲資料之格式（蔡政霖，2003）

地面光達所獲得之點雲資料應以掃描線點資料之型態儲存，因大部分地面光達作業過程均以水平或垂直方向進行掃描。

目前資料濾除的程序，大致上可分為：點資料亂點濾除、點資料的重整、點資料平滑化與點資料重新取樣，再由經處理過後的點資料中建構出曲線及曲面（莊峻超，2001）。

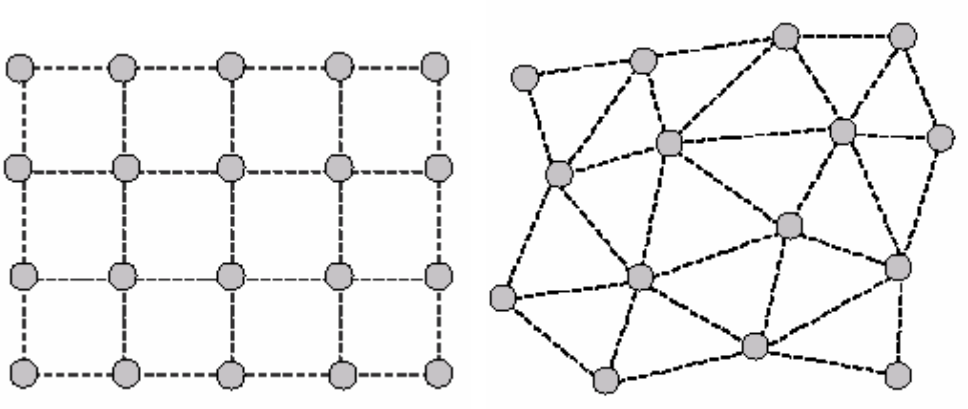
在資料結構化部分，大致可分為兩種處理方法，一為以規則網格分割的方式將點雲資料結構化；二為以不規則三角網（TIN, Triangular Irregular Network）的方法。

規則網格結構化是將三維座標內插成規則網格（如圖 2-2（a）），但內插後的規則網格會喪失部分空間資訊，其數據為衍生形成而非原始量測，對具三維特性的點雲資料而言，便失去原先可完整精確描述地物的特性。賴志恆（2003）以規則網格分割結合八分樹的方法，對點雲資料進行結構化之處理，有效地萃取出點雲資料中豐富的平面資訊，同時藉由八分樹建立的樹狀結構，連結點雲資料中隱含的平面資訊，並可由樹狀結構中建立點雲資料的三維空間索引，將點雲資料空間中的平面資訊予以結構化。結構化後的點雲資料由八分樹的分割或合併也能更清楚準確地展示實際點雲資料的空間資訊。完成結構化的點雲資料，在資料搜尋與處理上可提昇處理的速率，所萃取的平面資訊也可應用於三維地物重建。

而以不規則三角網法（如圖 2-2（b））來結構化點雲資料並進而建構三維模型的優點具有（陳俊諺，1999）：

a. 可以表現極為細緻的特徵。
b. 能完整的描述具有複雜幾何外形的模型。
c. 封閉的三角網格模型即構成實體。
d. 網格模型的處理大多為線性運算，不需要解複雜的非線性方程式，可增加處理的效率。
e. 在不同的軟體系統中，對三角網格模型的實體定義均相同，具有良好的流通性。

故在許多的應用領域中，如快速原型(RP, rapid product)、3D 動畫遊戲、虛擬實境等，都採用不規則三角網法做為模型建構的基礎，而建構三角網格的方法大多以狄勞尼三角形（Delaunay Triangulation）為理論基礎（陳俊諺，1999；蔡政霖，2003）。



（a）規則網格　　（b）不規則三角網法

圖 2- 2 點雲資料結構化

2-2 Scanalyze

本研究所採用之地面光達系統為 Optech ILRIS- 3D（Optech, 2004），其資料格式為 PIF 檔；而用以疊合分析之 Scanalyze 軟體則是以 PLY 檔作為處理格式。

Scanalyze 為 Stanford Computer Graphics Laboratory 所開發之自由軟體，為一種互動式電腦繪圖應用軟件，主要可以用以觀看、編輯、校準及合併距離影像並產生密集的 polygon mesh（SCGL, 2003）。已成功應用於數位化米開朗羅基羅像計畫，以雷射掃描儀將米開朗基羅雕像進行掃描後，得到表面的三維座標資訊，再以 Scanalyze 將其組合成一 3D 模型。

Scanalyze 主要可處理三種檔案格式：triangle-mesh PLY 檔（*.ply）、range-grid PLY 檔（*.ply）及 SD 檔（*.sd）。triangle-mesh PLY 檔是將一般的 triangle-mesh 的頂點任意連接且以列表方式儲存，而 range-grid PLY 檔及 SD 檔則是以規則的矩形陣列儲存點資料。此外，SD 檔也包含了描述掃描儀的幾何性質，如位置等；這些資訊在 Scanalyze 中直覺地可被用來推導不同的演算法。

當 PLY 檔或 SD 檔讀入後，可對所輸入的資料進行校準的動作，其中最有效的校準技術則為 ICP（Iterated Closest Point）法（Rusinkiewicz, 2001；Besl & Mckay, 1992），可將輸入的兩組資料選擇要以自動（或稱 All-Pairs Align）或手動的方式進行校準；也可以透過整體疊合（Global Registration）的方式，將校

準時產生的誤差平均地分散到校準的資料中，不至於造成校準時誤差的累積。而校準完新資料的位置及方位資訊，會以一 4x4 的矩陣方式儲存在*.xf 檔中。

由上述可知，SD 檔包含的是 range images 的資料，而 PLY 檔則可能包含了距離影像和一般的 mesh 資料，故 Scanalyze 中可處理這兩種類型的資料。當輸入的資料為距離影像時，在校準完之後，可利用 VRIP（volumetric range image processing）法（Curless, 1996）將資料合併，並組成一個具有位相關係且無縫（seamless）的多邊形 mesh，且僅能以 PLY 檔的格式儲存。

總結以上所言，Scanalyze 可提供上述功能：

1. 可對於 polygon meshes 及 range images 資料提供不同展示型態，如以實心體（solid）、魚網圖（wireframe）或點資料型態表示。
2. 提供 polygon mesh 的編輯及分析工具，如裁剪、複製、量測點位座標的功能。
3. 提供不同的校準方法，如 ICP、global registration。
4. 利用 VRIP 合併 range image。

2-3 PLY 檔

2-3-1 PLY 主要架構介紹

PLY（Polygon File Format）檔案主要是一連串的頂點及面所組成，並記錄每個頂點及面的資料型態及特性，如顏色、法向量等，以描述一個多邊形物體。

一個標準的 PLY 檔可定義為一群具有（x, y, z）三維座標之頂點的集合，並透過頂點來描述物體之表面，以”element”定義頂點及物體表面，PLY 格式之主要架構即是由每個 element 元素之列表所組成，而透過”property”詳細註明每個元素之型態及特性；簡言之，一個 PLY 檔案是由一群頂點及由頂點所描述的面所組成，並在 PLY 檔案中儲存每個點、線、面之型態及特性。除此之外，也可依使用者需求，自行建立新的 element 型態，並利用 property 來定義其屬性。新的 element 如 edges、cell（lists of pointers to faces）、materials 等（PLY，2004）。

element 及 property 之定義格式如下：

element <element-name> <number-in-file>

property <data-type> <property-name-1>

property <data-type> <property-name-2>

property <data-type> <property-name-3>

依上述之定義，先以 element 定義頂點數目，並利用 property 定義其資料型態及屬性值的排列順序（如依序定義 x, y, z），最後列出所有定義之頂點。謹以八個頂點作為例證：

element vertex 8
property float x
property float y
property float z
0 0 0
0 0 1
0 1 1
0 1 0
1 0 0
1 0 1
1 1 1
1 1 0

其中property可定義的資料型態如下表：

表 2- 1 property 定義之資料型態

Name	Type	Number of Bytes
char	character	1
uchar	unsigned character	1
short	short integer	2
ushort	unsigned short integer	2
int	integer	4
uint	unsigned integer	4
float	single-precision float	4
double	double-precision float	8

PLY檔案的主要架構如下：

檔頭（Header）

頂點列表（Vertex List）

面列表（Face List）

檔頭中描述檔案中所有元素之型態，包含元素名稱（如vertex）、檔案中元素總數目、及所有元素之屬性值；並說明檔案型態是ASCII或binary格式，其中comment通常為註解，以ply作為檔頭開頭，end_header作為檔頭結尾。

以下為一個cube之PLY檔範例：

```
ply
format ascii 1.0 { ascii/binary, format version number }
comment made by Greg Turk { comments keyword specified, like all lines }
comment this file is a cube
element vertex 8 { define "vertex" element, 8 of them in file }
property float x { vertex contains float "x" coordinate }
property float y { y coordinate is also a vertex property }
property float z { z coordinate, too }
element face 6 { there are 6 "face" elements in the file }
property list uchar int vertex_index { "vertex_indices" is a list of ints }
end_header { delimits the end of the header }
0 0 0 { start of vertex list }
0 0 1
0 1 1
0 1 0
1 0 0
1 0 1
1 1 1
1 1 0
4 0 1 2 3 { start of face list }
4 7 6 5 4
4 0 4 5 1
4 1 5 6 2
4 2 6 7 3
4 3 7 4 0
```

2-3-2 GRID 之檔案型態驗證

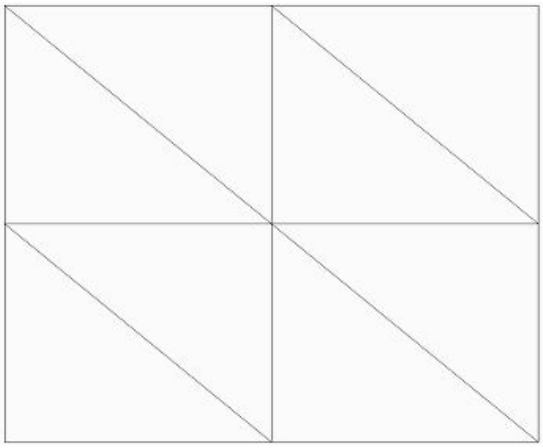
此處以Scanalyze軟體進一步說明PLY之檔案結構：

1. 檔頭定義column、row數目及grid之格子數目大小，中間之列表為每個頂點之x, y, z座標值，最後則註明在range grid中，若每個cell有點，則前面標註1，若無點存在則標註0。

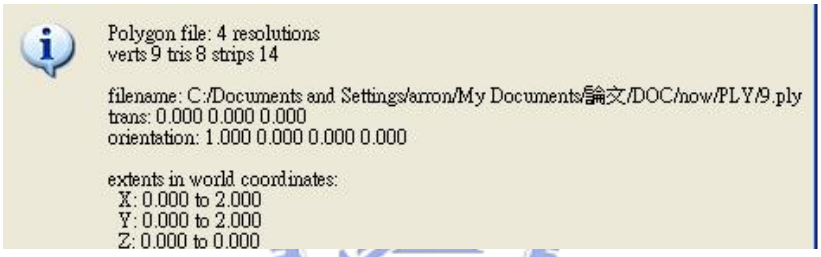
```
ply
format ascii 1.0
obj_info num_cols 3                    定義 column 數為 3
obj_info num_rows 3                    定義 row 數為 3
element vertex 9                       頂點總數為 9
property float x
property float y
property float z
element range_grid 9                   定義其 range grid 大小（cols × rows）
property list uchar int vertex_indices
end_header
0 0 0                                  { start of vertex list }
1 0 0
2 0 0
0 1 0
1 1 0
2 1 0
0 2 0
1 2 0
2 2 0
1 0
1 1
1 2
1 3
1 4
1 5
1 6
1 7
1 8
```

註明在 grid 中之位置，檔頭定義有九個 cell，若 cell 中有點，則前面標註 1，因此例中共有九個頂點，點數編號由 0~8。

此 PLY 檔在三維空間之示意圖（圖 2-3（a））及檔案資訊（圖 2-3（b））如下：



（a）PLY 之 Grid 型態



（b）PLY 資訊

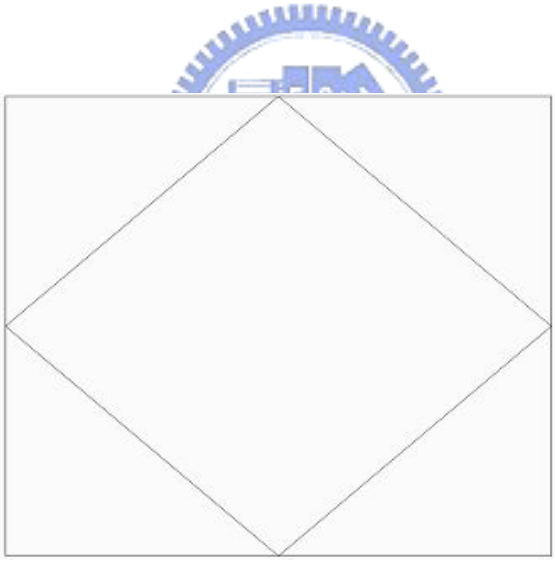
圖 2- 3 PLY 檔範例一

2. 將上例改寫如下： 將第五點去掉，前面註記改為0，總頂點數為8。

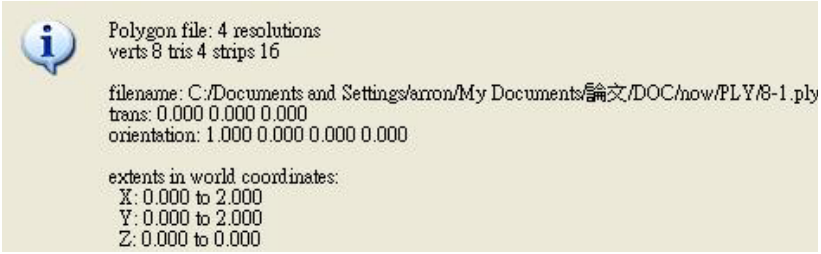
ply
format ascii 1.0
obj_info num_cols 3
obj_info num_rows 3
element vertex 8　　頂點總數為 8
property float x
property float y
property float z
element range_grid 9
property list uchar int vertex_indices
end_header
0 0 0
1 0 0

2 0 0
0 1 0
2 1 0
0 2 0
1 2 0
2 2 0
1 0
1 1
1 2
1 3
0 ←——— 去掉第五點
1 4
1 5
1 6
1 7

此 PLY 檔在三維空間之示意圖（圖 2-4（a））及檔案資訊（圖 2-4（b））如下：



（a）PLY 之 Grid 型態



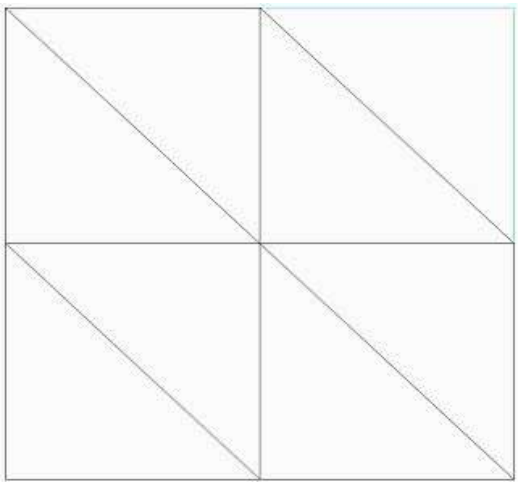
（b）PLY 資訊

圖 2-4 PLY 檔範例二

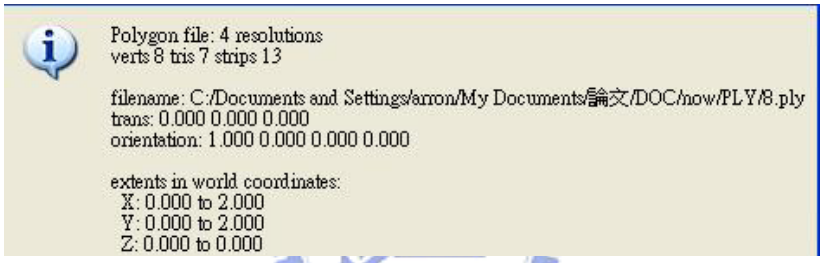
3. 上例改寫如下： 將第九點去掉，前面註記改為0，總頂點數為8。

ply
format ascii 1.0
obj_info num_cols 3
obj_info num_rows 3
element vertex 8　　頂點總數為 8
property float x
property float y
property float z
element range_grid 9
property list uchar int vertex_indices
end_header
0 0 0
1 0 0
2 0 0
0 1 0
1 1 0
2 1 0
0 2 0
1 2 0
1 0
1 1
1 2
1 3
1 4
1 5
1 6
1 7
0 ←——— 去掉第九點

此 PLY 檔在三維空間之示意圖（圖 2-5（a））及檔案資訊（圖 2-5（b））如下：



（a）PLY 之 Grid 型態



（b）PLY 資訊

圖 2- 5 PLY 檔範例三

2-3-3 小結

由 2-3-2 之實驗中可得到以下結論：

1. 由GRID之檔案型態範例一中可得知，每個vertex為下圖網格之頂點，而依其對角線劃分為八個三角形，以組成物體之表面，也驗證PLY檔案格式由一群頂點及由頂點所描述的面所組成，進而描述一多邊形物體。

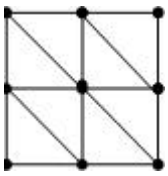


圖 2- 6 PLY 檔範例一

2. GRID檔案範例的第二種情形去掉中間之頂點，網格變成下圖所示，頂點數為

八，三角形數目為四。

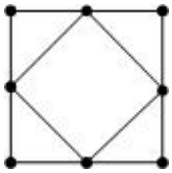


圖 2- 7 PLY 檔範例二

同理第三種情形則如下圖所示，頂點數仍為八，但三角形數目為七。

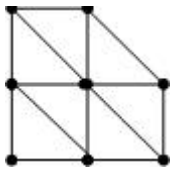


圖 2- 8 PLY 檔範例三

由此可知，PLY檔不因某頂點中沒有點存在，而無法組成三角形，會自行與最鄰近的頂點連結組成三角形，以描述物體表面形狀，但須在一個cell的距離內，超過此距離便無法組成三角形。

3. 由上述可知，PLY檔案會根據設定的range_grid大小，依序填入每個頂點，其排列順序如下所示，其中0~8則為PLY檔案中頂點列表排列之順序。

6 7 8

3 4 5

0 1 2

4. PLY檔案中所有的頂點在grid中應都有一個頂點位置來存放座標值，但該頂點位置也可能為空，即沒有頂點存在，此時最鄰近的三個頂點間會自行組成一三角形，如圖2-7、圖2-8所示。

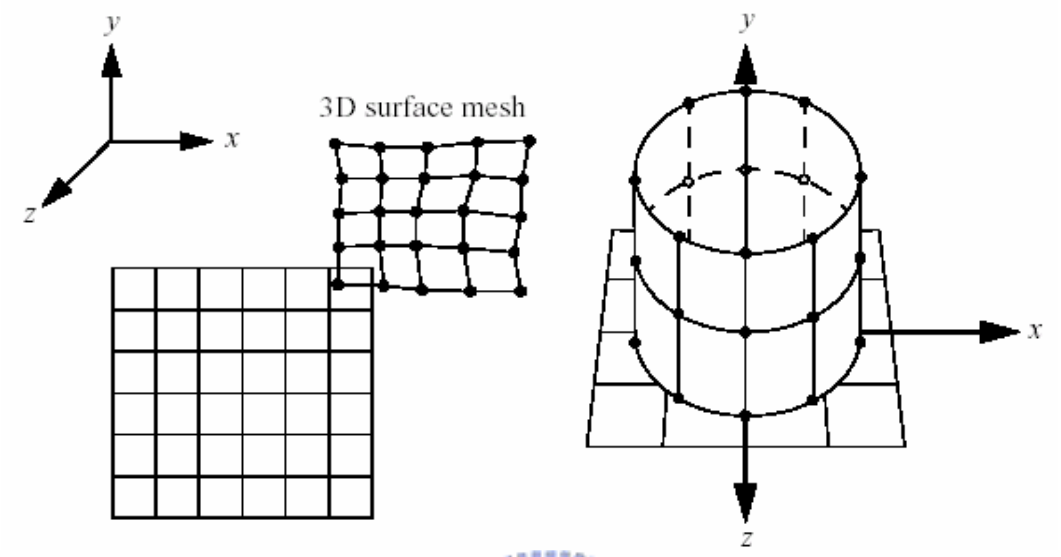
2-4 PIF 檔

2-4-1 參數影像之定義

PIF（Parametric Image Format）為一二進位之檔案格式，用以描述三維之參數影像（3D parametric images）。參數影像之定義為一三維表面網格（3D surface mesh），可以內插方式轉換至二維之參數空間中，PIF 檔同時也定義了平

面及圓柱之二維參數空間（InnovMETRIC, 2003）。

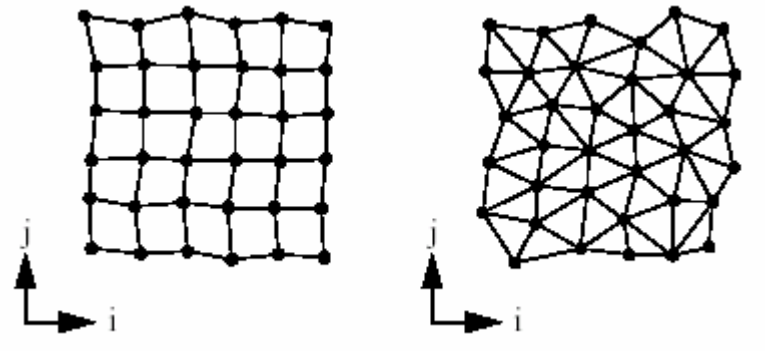
由三維表面網格所組成之平面（圖 2-9（a））及圓柱影像（圖 2-9（b）），可被轉換至二維參數空間，圖 2-9 為其轉換之示意圖。



（a）平面影像　　（b）圓柱影像

圖 2- 9 三維表面網格影像轉換至二維參數空間（InnovMETRIC, 2003）

原始參數影像（raw parametric image）之定義為經由三維影像獲取器所量測得到之三維表面網格，記錄了所有點之（x, y, z）座標值以及相鄰點間的資訊。在 PIF 檔案中，原始參數影像主要有兩種格式，一為將點資料以不規則陣列形式紀錄（如圖 2-10（a）），二為以多邊形網格的形式記錄（如圖 2-10（b））。



圖（a）不規則陣列　　（b）多邊形網格

圖 2- 10 原始參數影像（InnovMETRIC, 2003）

無論是以陣列形式或多邊形網格記錄，其最終目的均為將三維表面網格轉

換到一規則二維網格（如圖 2-11），並儲存所有點位資訊，此時被轉換後的參數影像稱為內插參數影像（interpolated parametric image）。

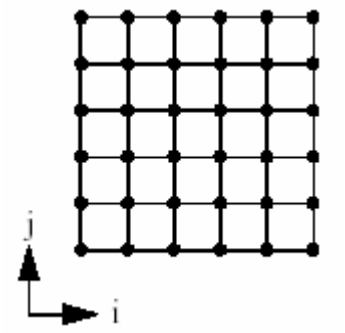


圖 2- 11 二維規則網格（InnovMETRIC, 2003）

2-4-2 PIF 檔座標系統及座標轉換

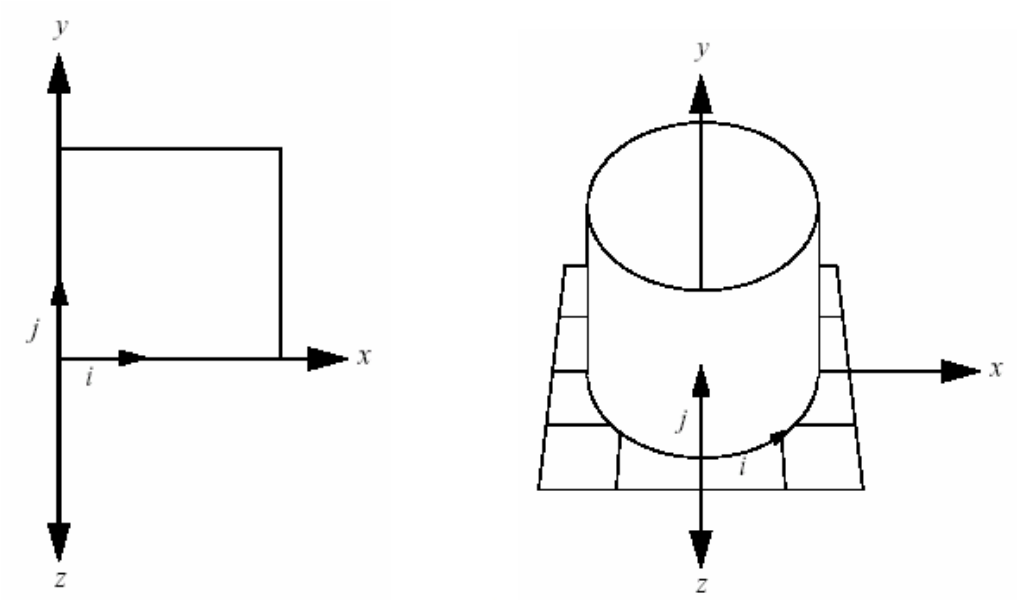
在 PIF 檔案格式中使用三種座標系統，分別為物座標系統（Data Coordinate System, DCS ）、中介座標系統（Intermediate Coordinate System, ICS）、內插參數影像座標系統（Interpolated Parametric Image Coordinate System, IPICS）。

物座標系統（DCS）為一標準的卡氏座標系統，即以（x, y, z）三維座標描述一原始參數影像。當以多個三維影像來描述同一個物體時，每幅影像都有其獨立之座標系統，故必須透過校準（Align, Register）的步驟，將所有之座標系轉換至同一個座標系統內，方能完整描述一物體。

原始三維影像在轉換到平面或圓柱之二維參數空間前，需先經剛體轉換，再以中介座標系統來記錄三維影像在空間中之位置；如若參數影像為平面型態，則 ICS 的 Z 軸代表平面之法向量；若為圓柱型態，則 ICS 的 Y 軸方向為圓柱之主軸。

內插參數影像座標系統（IPICS）為一固定比例的二維參數座標系統，用以定義平面或圓柱表面及描述內插參數影像，由三維卡式座標系轉換到二維的參數座標系的過程可稱為參數化（parameterization）。參數化的程序為將一點（x, y, z）轉換到二維參數空間中，改以（i, j）參數記錄，且可透過函數 f（i, j）之計算，反推求得（x, y, z）座標。所以若參數影像為平面型態，則（i, j）=（x, y），

f(i, j)= z(如圖 2-12(a))。若為圓柱型態，則 $i = a\tan\left(\frac{x}{z}\right)$，j = y，$f(i,j) = \sqrt{x^2 + y^2}$（如圖 2-12（b））。



（a）　　（b）

圖 2- 12 參數化示意圖（a）平面型態（b）圓柱型態（InnovMETRIC, 2003）

IPICS 主要目的為透過規則網格的儲存及函數 f 的計算，有效地描述內插參數影像，以（i, j）參數來記錄原始參數影像的資訊；在規格網格中，無論水平或垂直方向，鄰近的參數距離均為 1，其座標原點位於左下角，如圖 2-13 所示：

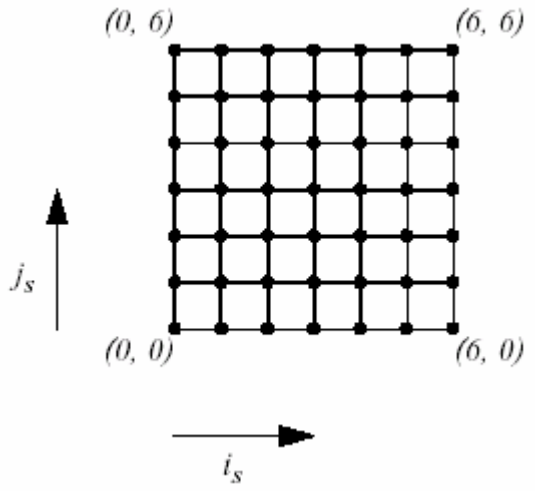


圖 2- 13 IPICS 座標系統（InnovMETRIC, 2003）

故全部座標轉換的過程可以下列的流程圖來表示（假設原始參數影像為平面型態）：

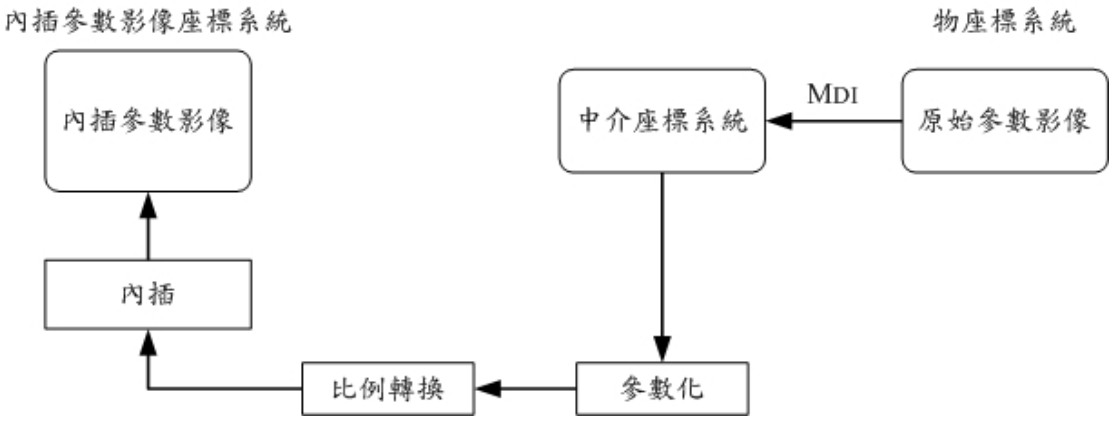


圖 2- 14 座標轉換流程（InnovMETRIC, 2003）

其中 M_{DI} 為一 4x4 的轉換矩陣，表示如下：

$$M_{DI} = \begin{bmatrix} r_1 & r_2 & r_3 & t_x \\ r_4 & r_5 & r_6 & t_y \\ r_7 & r_8 & r_9 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2\text{-}1)$$

故由 DCS 轉換到 ICS 可以下式表示：

$$\begin{bmatrix} x_{ICS} \\ y_{ICS} \\ z_{ICS} \\ 1 \end{bmatrix} = M_{DI} \begin{bmatrix} x_{DCS} \\ x_{DCS} \\ x_{DCS} \\ 1 \end{bmatrix} \qquad (2\text{-}2)$$

在（i, j）之參數座標系中，具有兩個尺度因子 i_scale 及 j_scale 來調整參數尺度大小，故：

$$\begin{aligned} i_s &= i / i\_scale \\ j_s &= j / j\_scale \end{aligned} \qquad (2\text{-}3)$$

整個座標轉換流程為可逆，故由 ICS 轉換到 DCS 可寫成：

$$\begin{bmatrix} x_{DCS} \\ y_{DCS} \\ z_{DCS} \\ 1 \end{bmatrix} = M_{ID} \begin{bmatrix} x_{ICS} \\ x_{ICS} \\ x_{ICS} \\ 1 \end{bmatrix} \quad (2\text{-}4)$$

$$\begin{aligned} i &= i \cdot i\_scale \\ j &= j \cdot j\_scale \end{aligned} \quad (2\text{-}5)$$

此時 ICS 中的（x, y, z）分別如下：

$$\begin{aligned} x &= i \\ y &= j \\ z &= f(i, j) \end{aligned} \quad \text{（參數影像為平面型態）} \quad (2\text{-}6)$$

$$\begin{aligned} x &= f(i, j) \cdot \sin i \\ y &= j \\ z &= f(i, j) \cdot \cos i \end{aligned} \quad \text{（參數影像為圓柱型態）} \quad (2\text{-}7)$$

其示意圖如下：

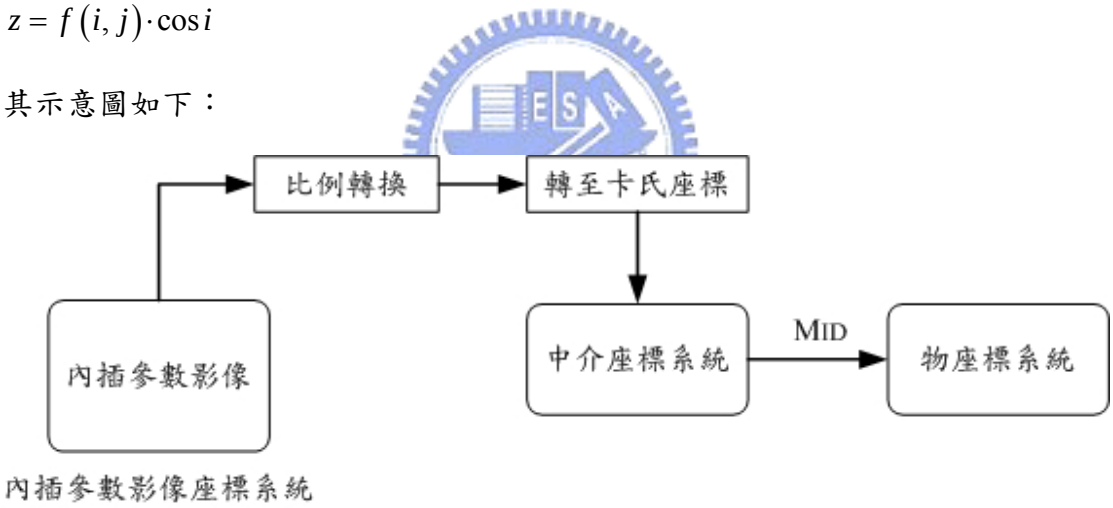


圖 2- 15 座標轉換流程（逆向）（InnovMETRIC, 2003）

2-4-3 PIF 檔案結構

PIF 檔案主要由三個部分所組成，第一部份為檔頭（header），包含關於檔案結構的資訊；第二部分為記錄三維影像的資訊；第三部分則為非必要之部分，如顏色等資訊。

檔頭部分主要說明 PIF 檔案的架構及定義所記錄的相關資訊，如 PIF 檔案版本、陣列大小、（i、j）參數的尺度大小、轉換矩陣及三維影像獲取器的座標位置。標準檔頭格式為如下：

```
struct header
{
char format_version[64]；
char user_comments[128]；
char dummy1[8]；
long image_param_flag；
long image_data_type；
float invalid_point；
long array_width；
long array_height；
long data_block_length；
long scale_flag；
float i_scale；
float j_scale；
long transfo_matrix_flag；
double transfo_matrix[16]；
long image_color_flag；
long color_block_length；
long camera_position_flag；
float camera_x；
float camera_y；
float camera_z；
long dummy2[30]；
}；
```

檔頭詳細定義請參見附錄一

第二部分為 3D data 部分，在檔頭定義中有一 image_data_type 之識別字，若 image_data_type 被設定為 0，則這個 data block 被用來記錄 IPICS 中之二維陣列資訊並用以描述內插參數影像；二維陣列的大小也同時在檔頭資料中定義，陣列的寬度及高度分別定義為 array_width 及 array_height。下面以一個 4x4 的二維陣列作為例證，座標原點為左下角，由左到右，由下而上記錄陣列元素：

12 13 14 15

8 9 10 11

4 5 6 7

0 1 2 3

若 image_data_type 設定為 1，則 data block 以二維陣列的形式記錄 DCS 中的三維座標（x, y, z），以用來描述原始參數影像，每個陣列元素包含（x, y, z）座標以描述一個三維點位，同樣由左到右，由下而上記錄陣列元素。

若 image_data_type 設定為 2，則 data block 包含兩個字串（strings）。第一個字串主要紀錄用來描述三維影像表面的外部多邊形檔案之路徑；第二個字串則記錄對應多邊形（corresponding polygon）的檔案格式。

第三部分則為非必要的顏色資訊，主要在檔頭中的 image_color_flag 定義。

image_color_flag = 0　PIF 檔中無顏色資訊

image_color_flag = 1　PIF 檔為灰階

image_color_flag = 3　PIF 檔具有 RGB 三種顏色資訊