附錄一 PIF 檔頭定義

format_version： A 64-byte field reserved for an identifier string which gives the name of the format and the version number. InnovMetric currently writes “PIF Format v2.0” in this field.

user_comments： A 128-byte field where you may write any information about the model.

dummy1： An 8-byte field that may be used for special characters.

image_param_flag： A flag indicating the type of image parameterization.

If 0, the parameterization is PLANAR.

If 1, the parameterization is CYLINDRICAL.

image_data_type： A flag indicating the type of data written in the 3D data block：

0 – the data block contains a 2D array of floating-point numbers expressed in the IPICS. This data type describes a parametric image that is already interpolated.

1 – the data block contains a 2D array of *(x, y, z)* coordinates expressed in the DCS. This data type describes a raw parametric image. Connectivity between 3D points is derived from the connectivity in the array.

2 – the data block contains two strings specifying a) the path to an external polygonal file, and b) the corresponding polygonal file format.

invalid_point： If image_data_type is 0, an array element whose value is equal to nvalid_point should be considered as an invalid point.

If image_data_type is 1, a point whose *z* coordinate is equal to invalid_point should be considered as an invalid point.

array_width： Width of the 2D array (not used if the data is contained in an external polygonal file).

array_height： Height of the 2D array (not used if the data is contained in an external polygonal file).

data_block_length： Length in bytes of the data block.

If image_data_type is 0 or 1, the data block length is equal to the size of the 2D array in bytes.

If image_data_type is 2, data_block_length should be equal to 1024 bytes.

scale_flag： A flag related to the scaling factors：

0 – the scaling factors should not be used.

1 – the scaling factors should be used. If image_data_type is 0, the scaling factors must be specified. If image_data_type is 1 or 2, the scaling factors are optional.

i_scale： Scaling factor for the *i* coordinate. If the parameterization is PLANAR, i_scale is specified in image units. If the parameterization is CYLINDRICAL, i_scale is specified in degrees.

j_scale： Scaling factor for the *j* coordinate, specified in image units. If the parameterization is PLANAR, i_scale and j_scale should be equal.

transfo_matrix_flag： A flag indicating the transformation performed between the DCS and the ICS：

0 – the transformation matrix between the DCS and the ICS is an identity matrix. In this case, the DCS and the ICS are equivalent and the data contained in field transfo_matrix is not

used.

1 – the field transfo_matrix specifies a matrix *MDI* that transforms points from the DCS to the ICS.

2 – the field transfo_matrix specifies a matrix *MID* in order to transform points from the ICS to the DCS.

transfo_matrix： Sixteen matrix elements specified in double-precision format. The matrix elements *mi* are specified from left to right and from top to bottom..

image_color_flag： A flag indicating color information. Color information is optional in a PIF file. If the image data is specified in an external polygonal file, this flag should be set to 0.

0 – there is no color information in the file.

1 – the color block specifies a grey level for each array element in the data block.

3 – the color block specifies RGB colors for each array element in the data block.

4 – the color block specifies RGBA colors for each array element in the data block.

color_block_length： Length in bytes of the color block.

camera_position_flag：

A flag related to the *(x, y, z)* position of the 3D digitizer in the DCS：

0 – no camera position is specified.

1 – a camera position is specified by camera_x, amera_y,camera_z.

camera_x： *x* position of the 3D digitizer in the DCS.

camera_y： *y* position of the 3D digitizer in the DCS.

camera_z： *z* position of the 3D digitizer in the DCS.

dummy2： Reserved for future use.

附錄二 xyz 座標轉 PLY 檔程式碼

```
// convert xyz to ply

#include <stdio.h>
#include <math.h>
int main()
{
	// 0.Initialize
	float x,y,z；
	float maxX,maxY, maxZ,minX,minY, minZ；
	int j=0,row=0；
	FILE *cfptr1；
	FILE *cfptr2；
	FILE *cfptr3；
	FILE *cfptr4；

	cfptr1=fopen("input.txt", "r")；
	cfptr2=fopen("2.txt", "w+")；
	cfptr3=fopen("3.txt", "w+")；
	cfptr4=fopen("out.ply", "w+")；

	// 1. find the min and max of X and Y
	fscanf(cfptr1,"%f%f%f", &maxX, &maxY, &maxZ)；
	row++；
	minX = maxX；
	minY = maxY；
	minZ = maxZ；

	while(!feof(cfptr1))
	{
		fscanf(cfptr1,"%f%f%f", &x, &y, &z)；
		if ( x>maxX )
			maxX=x；
		if ( y>maxY )
			maxY=y；
		if ( z>maxZ )
			maxZ=z；
```

```
        if ( x<minX )
                minX=x；
        if ( y<minY )
                minY=y；
        if ( z<minZ )
                minZ=z；

        row=row+1；
}

// 2. Find the min distance between points
float fXd = 65535；
float fYd = 65535；
float fTempYd = 0.0；
float fTempXd = 0.0；
float fNowX；
float fNowY；
float fNowZ；
float fTempX；
float fTempY；
float fTempZ；
int nNowRow = 0；

for ( nNowRow = 0； nNowRow < row； nNowRow++)
{
        rewind(cfptr1)；
        // find the current row ： x y z
        for (   int i = 0； i <= nNowRow； i++)
        {
                fscanf(cfptr1,"%f%f%f", &fNowX, &fNowY, &fNowZ)；
        }

        // find min Xd
        rewind(cfptr1)；
        while(!feof(cfptr1))
        {
```

```
            fscanf(cfptr1,"%f%f%f", &fTempX, &fTempY, &fTempZ)；
            if ( fNowX != fTempX )
            {
                fTempXd = fTempX - fNowX；
                if (fTempXd < 0)
                {
                    fTempXd *= -1 ；
                }

                if ( fTempXd < fXd )
                {
                    fXd = fTempXd ；
                }
            }
        }
}

for ( nNowRow = 0 ；  nNowRow <= row ；  nNowRow++)
{
        rewind(cfptr1)；
        for ( int j = 0 ；  j < nNowRow ；  j++)
        {
            fscanf(cfptr1,"%f%f%f", &fNowX, &fNowY, &fNowZ)；
        }

        // find min Yd
        rewind(cfptr1)；
        while(!feof(cfptr1))
        {
            fscanf(cfptr1,"%f%f%f", &fTempX, &fTempY, &fTempZ)；
            if ( fNowY != fTempY )
            {
                fTempYd = fTempY - fNowY ；
                if (fTempYd < 0)
                {
                    fTempYd *= -1 ；
                }
```

```
                if ( fTempYd < fYd )
                {
                    fYd = fTempYd；
                }
            }
        }
}

// 3. Compute the grid number
int nGridCol = int( ( maxX - minX ) / fXd ) + 1；
int nGridRow = int( ( maxY - minY ) / fYd ) + 1；

// 4. Place points into grid
float fGridXL；
float fGridXR；
float fGridYB；
float fGridYT；
int flag；
float fGX；  // the point in grid
float fGY；
float fGZ；
int nGridCount = 0；

for ( int i = 0；  i < nGridRow  ；  i++)   //i represent y
{
    fGridYB = minY + i * fYd；
    fGridYT = minY + (i + 1) * fYd；

    for ( int j = 0；  j < nGridCol  ；  j++)   //j represent x
    {
    fGridXL = minX + j * fXd；
    fGridXR = minX + (j + 1) * fXd；

        rewind(cfptr1)；
        flag = 0；
        while(!feof(cfptr1))
        {
            fscanf(cfptr1,"%f%f%f", &fTempX, &fTempY, &fTempZ)；
```

```
                    if (( fTempX < fGridXR && fTempX >= fGridXL) &&
( fTempY < fGridYT && fTempY >= fGridYB ))
                    {
                        flag = 1 ;
                        fGX = fTempX ;
                        fGY = fTempY ;
                        fGZ = fTempZ ;
                    }
                }

                if ( flag == 1 )
                {
                    fprintf(cfptr3,"\n1 %d", nGridCount) ;

                    if ( i == ( nGridRow - 1) && j == ( nGridCol - 1))
                        fprintf(cfptr2,"%f %f %f", fGX, fGY, fGZ) ;
                    else
                        fprintf(cfptr2,"%f %f %f\n", fGX, fGY, fGZ) ;

                    nGridCount ++ ;
                }else
                {
                    fprintf(cfptr3,"\n0") ;
                }

            }
        }
char aaa = EOF ;
fprintf(cfptr2,"%c", aaa) ;
fprintf(cfptr3,"%c", aaa) ;

rewind(cfptr2) ;
rewind(cfptr3) ;

fprintf(cfptr4,"ply\n") ;
fprintf(cfptr4,"format ascii 1.0\n") ;
fprintf(cfptr4,"obj_info num_cols %d\n", nGridCol) ;
fprintf(cfptr4,"obj_info num_rows %d\n", nGridRow) ;
```

```
fprintf(cfptr4,"element vertex %d\n", row)；
fprintf(cfptr4,"property float x\n")；
fprintf(cfptr4,"property float y\n")；
fprintf(cfptr4,"property float z\n")；
fprintf(cfptr4,"element range_grid %d\n", (nGridCol*nGridRow))；
fprintf(cfptr4,"property list uchar int vertex_indices\n")；
fprintf(cfptr4,"end_header\n")；

char cT；
while( !feof(cfptr2))
{
	fscanf(cfptr2, "%c", &cT)；
	fprintf(cfptr4,"%c", cT)；
}

while(!feof(cfptr3))
{
	char cT；
	fscanf(cfptr3, "%c", &cT)；
	fprintf(cfptr4,"%c", cT)；
}
fclose(cfptr1)；
fclose(cfptr2)；
fclose(cfptr3)；
fclose(cfptr4)；
return 0；
}
```

附錄三 將散佈的三維點位轉換成 PLY 格式

在 2-3 中曾說明過有關 PLY 之檔案格式，在本節中以一散佈的三維點位來進一步說明 PLY 詳細的檔案結構。

一、程式流程

假設有一群散佈之三維點位，依下列步驟將散佈點轉換成 PLY 檔。

1. 先由散佈點中找出 X、Y 座標的最大最小值，以求出最小包圍四邊形。
2. 求點與點間之最短距離，作為切割 cell 的邊界長度。
3. 計算網格數，range_grid=cols × rows
4. 利用 2 中所定義的每個 cell 之大小，判斷是否有點落於 cell 中，若有點則輸出 1 並記錄頂點之排列編號；若 cell 中沒有點，則輸出 0。
5. 輸出 PLY 檔。

二、實例說明

A. 範例一：輸入一散佈的三維點如下

2 0 0
0 0.5 0
0 1 0
1 1 0
2 1.5 0
0 2 0
1 2 1
2 2 0

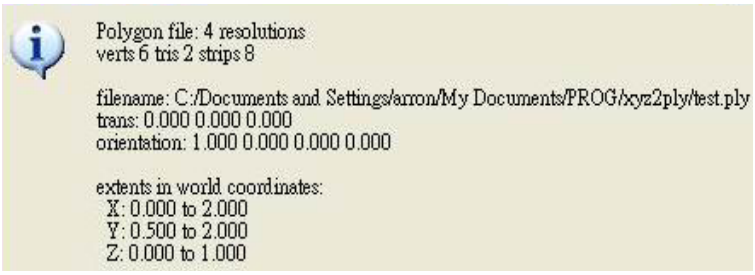
經轉換成 PLY 檔如下：

ply
format ascii 1.0
obj_info num_cols 3
obj_info num_rows 5
element vertex 8
property float x
property float y
property float z
element range_grid 15
property list uchar int vertex_indices

end_header
2 0 0
0 0.5 0
0 1 0
1 1 0
2 1.5 0
0 2 0
1 2 1
2 2 0
0
0
1 0
1 1
0
0
1 2
1 3
0
0
0
1 4
1 5
1 6
1 7

此 PLY 檔在三維空間之示意圖如下：

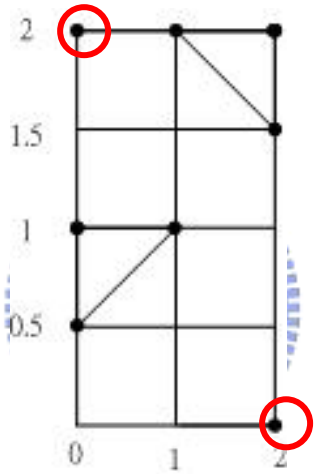
（a）三維空間示意圖



（b）檔案資訊

圖一 轉換範例一

故輸入之散佈點位經轉檔後，改以二維陣列方式儲存頂點位置，座標原點為左下角，分別由左至右，由下而上記錄（注意此處的 1 為第一個頂點，與 PLY 中第一個頂點為 0 不同，以方便區別）：



（a）Gird 型態儲存頂點

6 7 8

0 0 5

3 4 0

2 0 0

0 0 1

（b）頂點排列編號

圖二 二維陣列（紅色圈選點代表無法組成三角形之點）

由圖一（b）中可知，此 PLY 檔在 Scanalyze（SCGL, 2003）中只出現兩個三角形，且只記錄六個頂點，並非所輸入的八個，由此可推知，Scanalyze 只記錄能組成三角形之頂點，無法組成三角形的頂點，會自動去掉，由上圖 Y 值的

分佈範圍也可驗證這個結論，Y 值範圍為：0.500~2.000，並非輸入範圍 0.00~2.00。

B. 範例二：改寫範例一之散佈的三維點如下

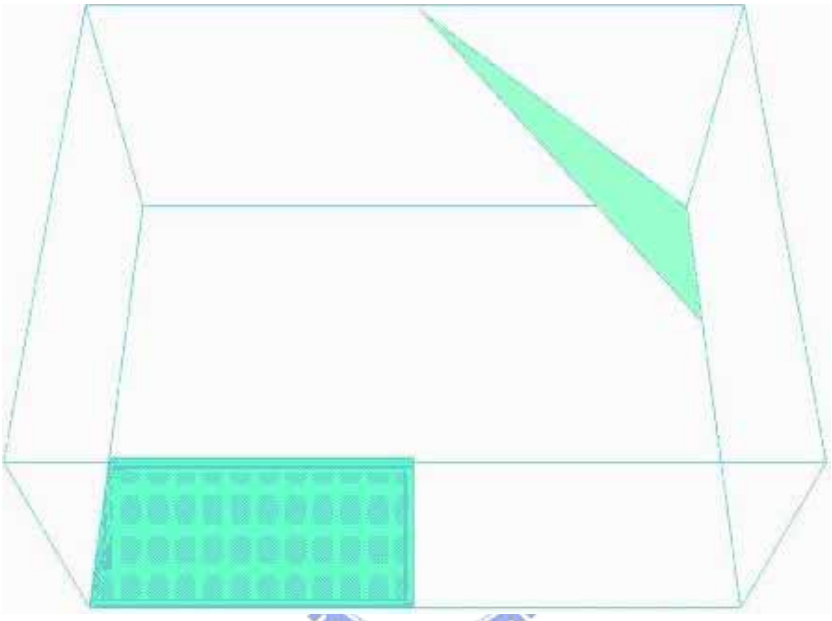
0 0.5 0
1 0.5 0
0 1 0
1 1 0
2 1.5 0
0 2 0
1 2 1
2 2 0

經轉換成 PLY 檔如下：

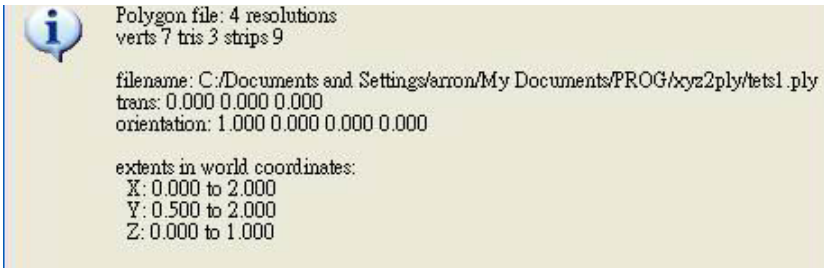
ply
format ascii 1.0
obj_info num_cols 3
obj_info num_rows 4
element vertex 8
property float x
property float y
property float z
element range_grid 12
property list uchar int vertex_indices
end_header
0 0.5 0
1 0.5 0
0 1 0
1 1 0
2 1.5 0
0 2 0
1 2 1
2 2 0
1 0
1 1
0
1 2
1 3

0
0
0
1 4
1 5
1 6
1 7

其 PLY 檔在三維空間之示意圖如下：



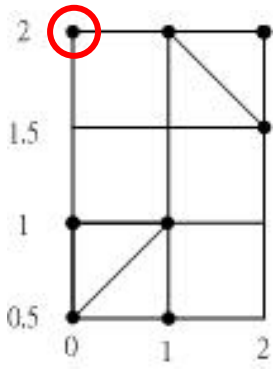
（a）三維空間示意圖



（b）檔案資訊

圖三 轉換範例二

其儲存陣列示意圖如下；



（a）Gird 型態儲存頂點

6　7　8

0　0　5

3　4　0

1　2　0

（b）頂點排列編號

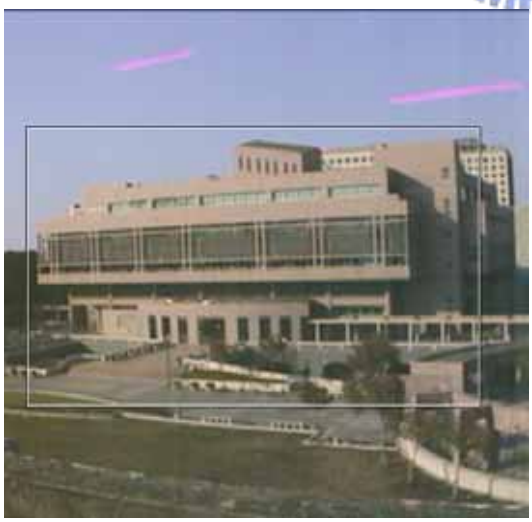
圖四　二維陣列（紅色圈選點代表無法組成三角形之點）

由圖三（b）可得知，此 PLY 檔在 Scanalyze 中出現三個三角形，且記錄七個頂點，並非所輸入的八個，同上之推論，Scanalyze 只記錄能組成三角形之頂點，並同時符合最小外包四邊形之原則。

附錄四 成大圖書館疊合之研究

一、實驗區資料說明

在成大圖書館前架設四個不同測站進行掃瞄作業，其中測站一及測站二分別架設於圖書館大門口左右兩側前方草地上。測站三及測站四，此兩測站選擇架於測量系樓頂進行掃瞄作業，主要為了考量避免樹木之遮蔽，詳細資料數據如表一。

<table>
<tr><td>  </td><td>  </td></tr>
<tr><td>測站一</td><td>測站二</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>測站三</td><td>測站四</td></tr>
</table>

圖一 各測站之掃描影像

表一 各測站掃描資料數據

測站編號	掃描距離 (m)	掃描線總數	每條掃描線之掃描點總數	掃描資料大小（MB）	掃描點取樣間隔 (mm)
一	75	906	1405	15.7	35
二	83	781	1362	13.1	39
三	107	1299	1840	29.6	34
四	117	1250	1954	30.2	37

二、實驗流程

A. 選取共軛點

以建築物之屋角為共軛點進行連結。

B. 以 ICP 法進行疊合

與竹東崩塌整治地之處理方式相同。

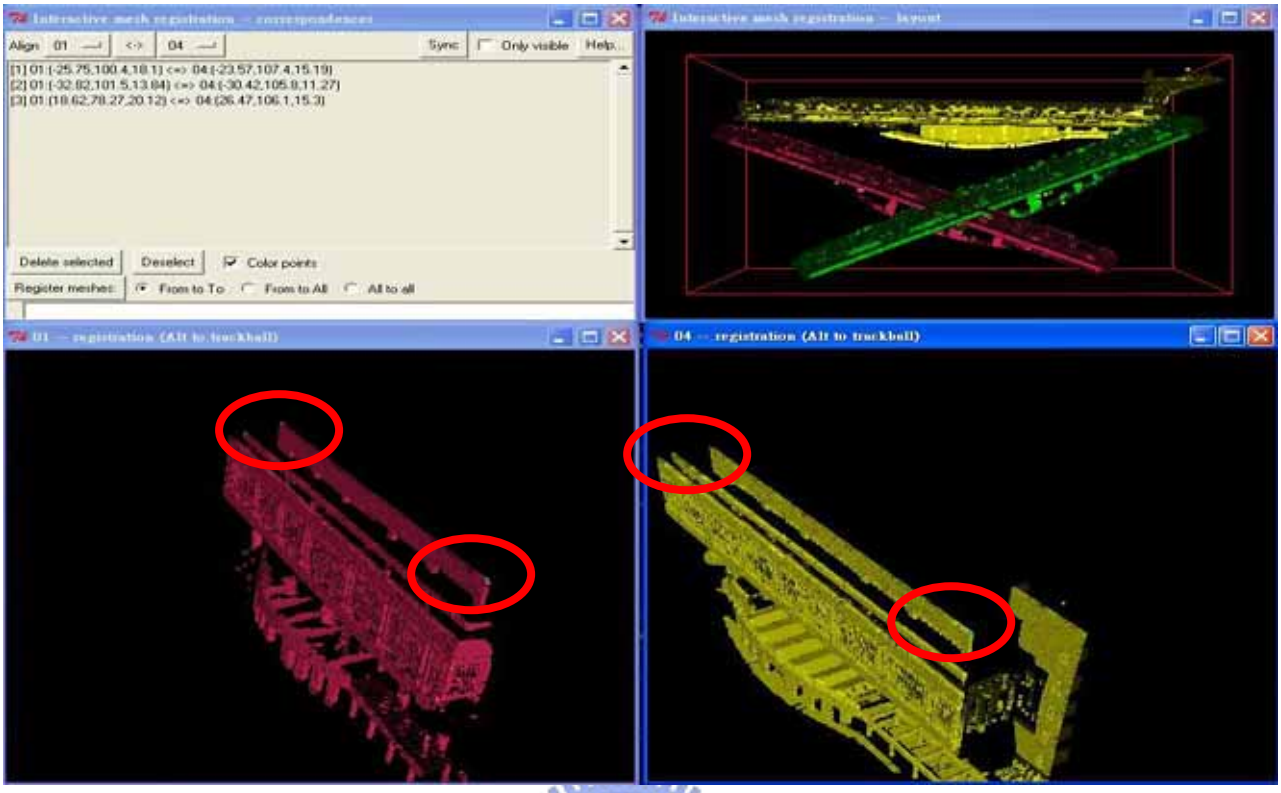
C. 整體疊合（Global Registration）

不以兩個測站為一組的方式，將點雲資料逐一進行疊合，而是將全部測站資料進行合併的動作，求其最佳轉換關係，盡量避免可能的疊合誤差的累積；此外，將兩組相鄰測站之疊合成果作為整體疊合時的約制，有效地將疊合誤差均勻地散佈在每個測站間，避免掉入區域最小值。

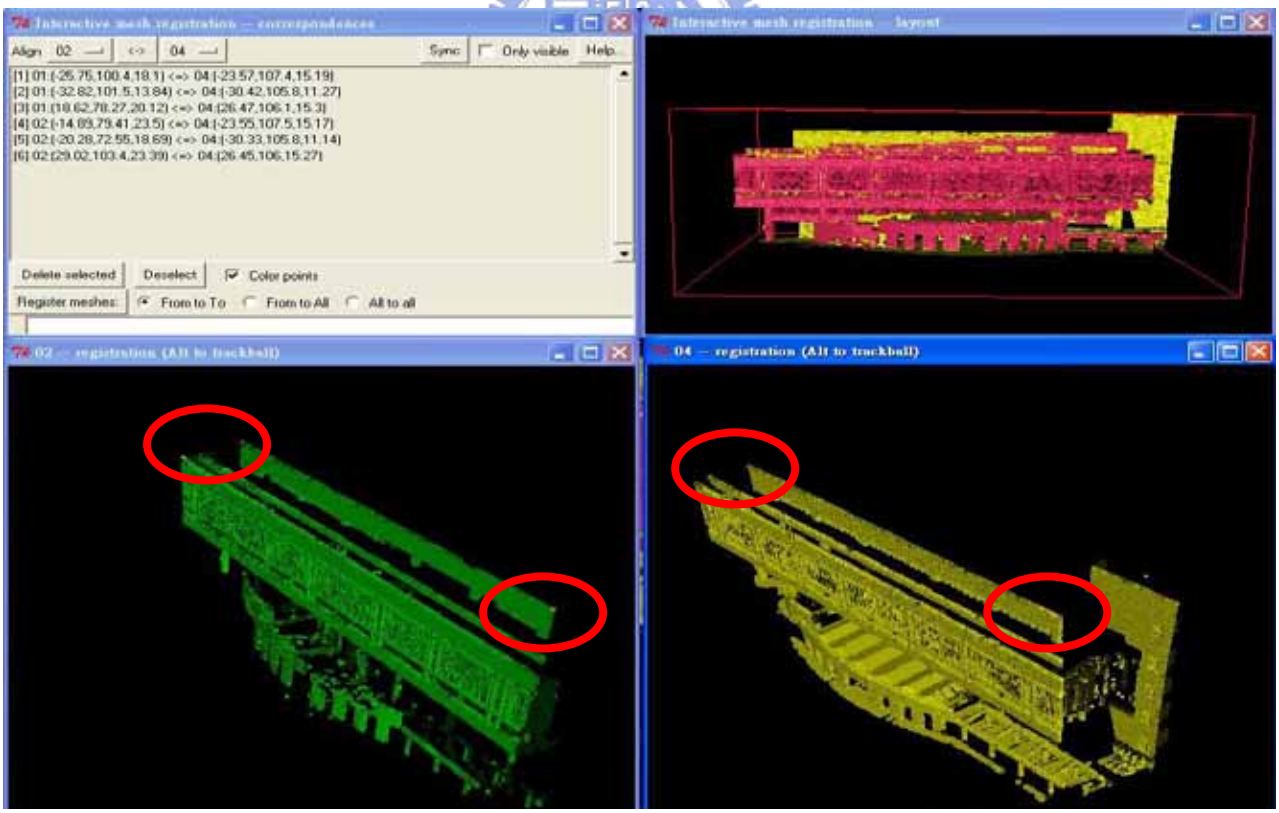
三、實驗成果

A. 共軛點選取

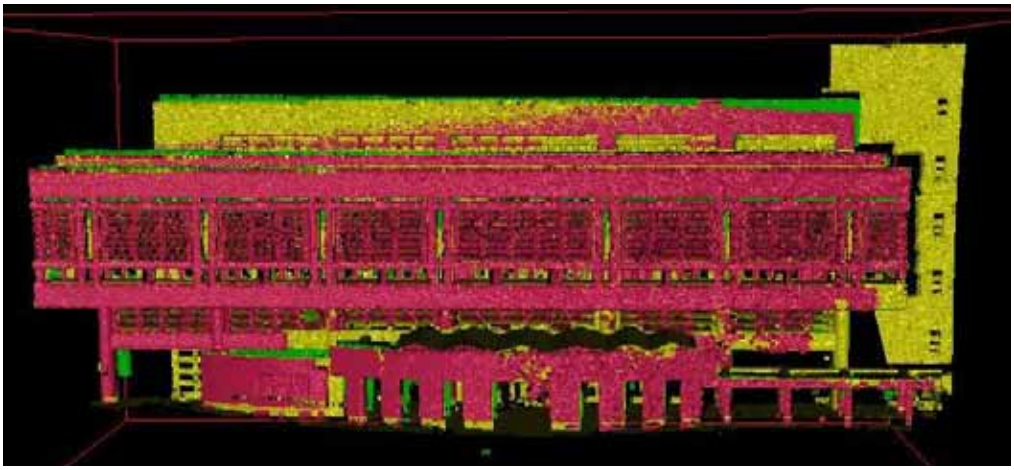
如圖二、圖三所示，測站一資料以紅色，測站二資料以綠色，測站四則以黃色表示。此處以測站四為固定不動之基準，分別選取測站一、測站二分別與測站四之控制點，以進行點雲連結。



圖二 選取測站一與測站四之共軛點



圖三 選取測站二與測站四之共軛點



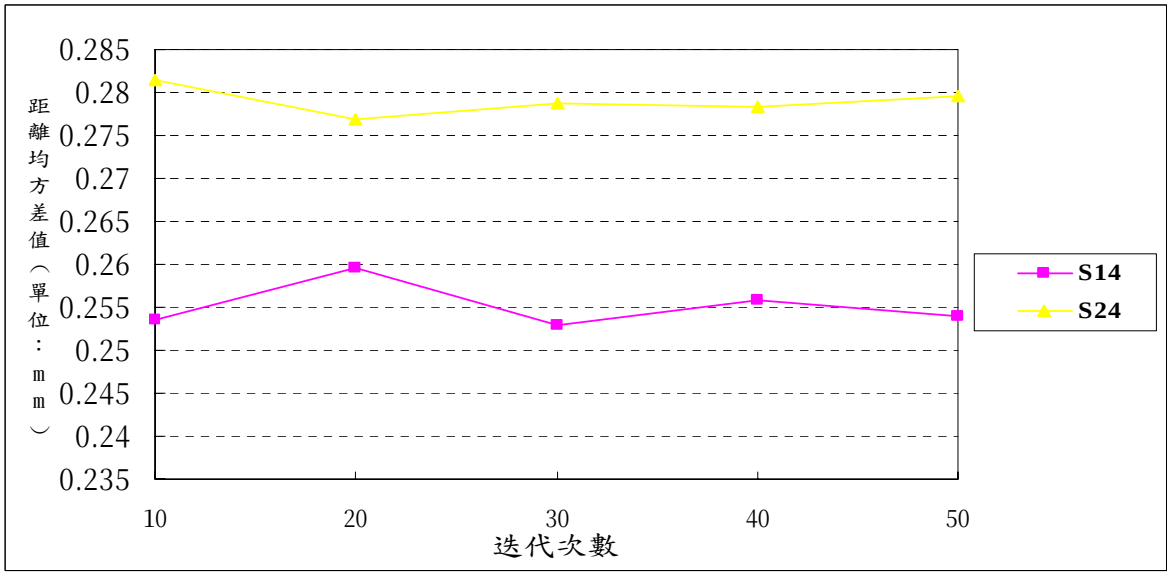
圖四 三測站之點雲連結成果

B. 疊合精度比較表

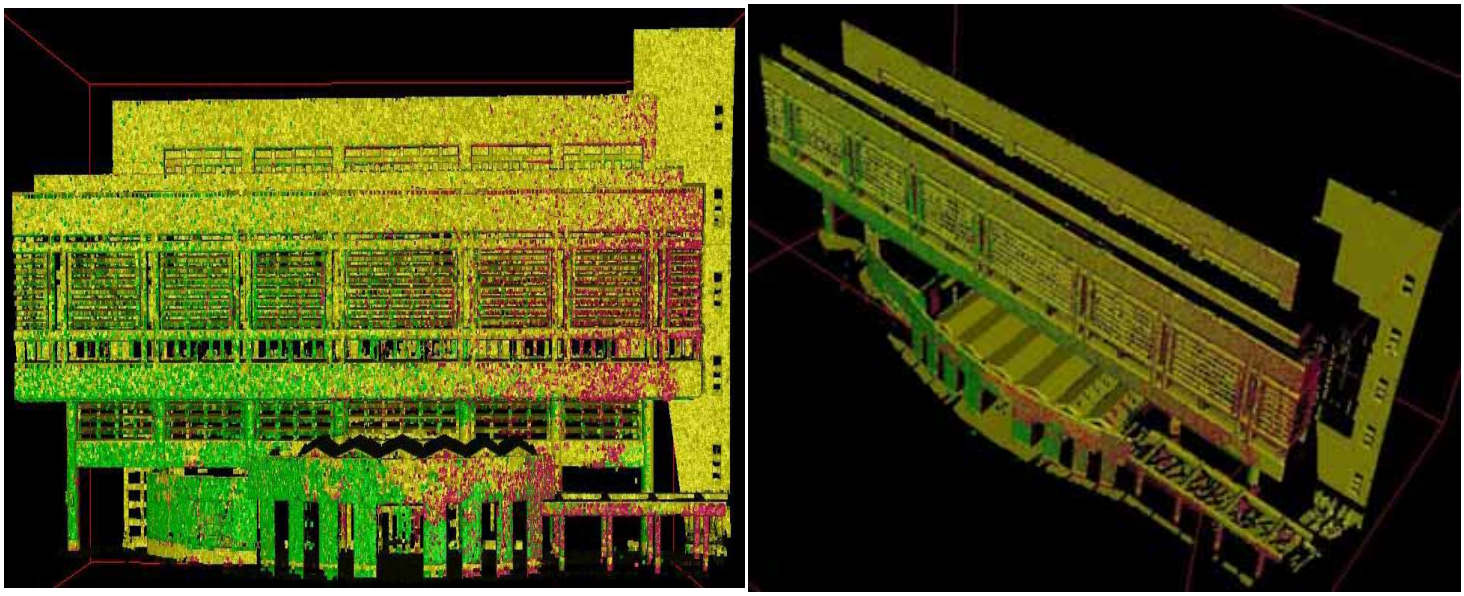
表二為 3 個測站間的疊合精度比較表，測站一、二分別以測站 4 為基準進行疊合動作。

表二 不同測站疊合過程之距離均方差值（單位：mm）

迭代次數 / 疊合測站	10	20	30	40	50
測站一◄►測站四	0.253449	0.259643	0.252833	0.255777	0.254031
測站二◄►測站四	0.281497	0.276943	0.278682	0.278304	0.279567



圖五 測站 1、2、4 疊合距離均方差值統計圖



正視圖　　等角透視圖

圖六 三測站之疊合成果圖

C. 整體疊合成果

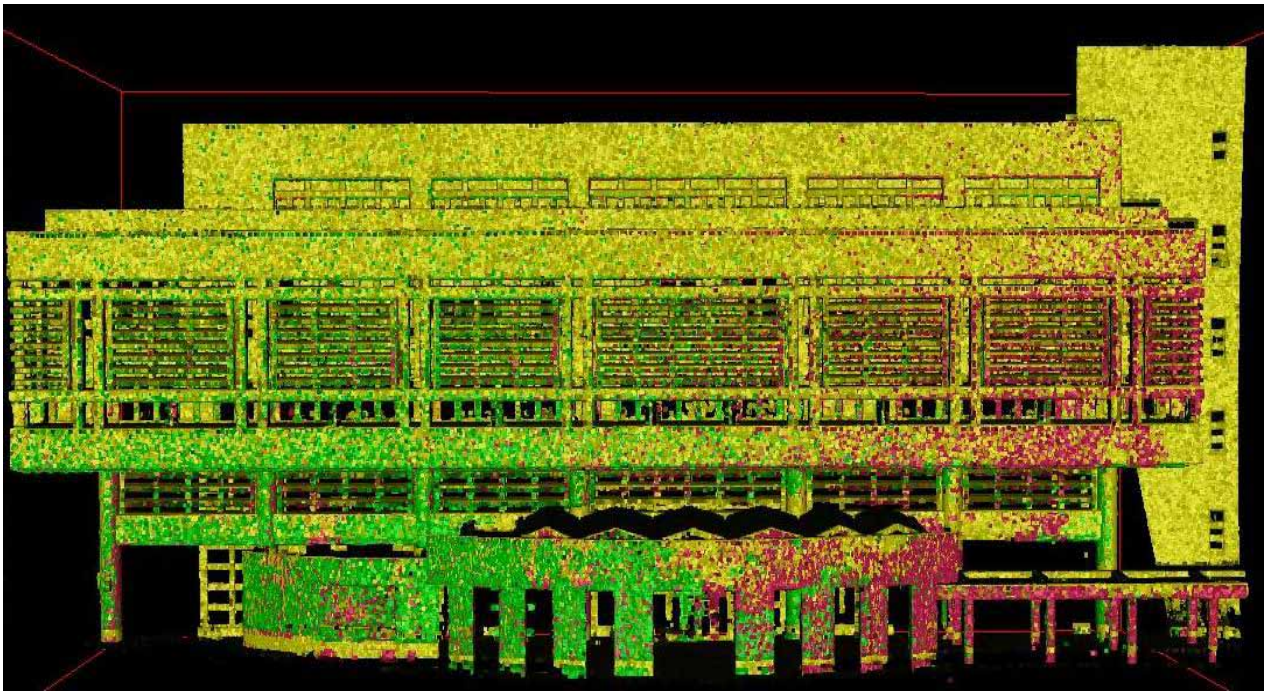
以上述測站一與測站四，測站二與測站四初步疊合之成果作為整體疊合時的約制，即分別以測站一與測站四之疊合誤差（0.254mm），測站二與測站四之疊合誤差（0.280mm）作為約制，進行整體疊合，當前後兩次迭代的距離均方差值小於一門檻值τ（τ=0.01mm），則停止迭代。

表三 各測站疊合誤差比較表

		點對點疊合誤差（單位：mm）		
測站編號	疊合總數	最小值	平均值	最大值
測站一	2	0.242	0.248	0.254
測站二	2	0.242	0.261	0.280
測站四	2	0.254	0.267	0.280

表四 整體疊合成果比較表

測站編號	對應測站	RMS	點到點之距離均方差	點到平面之距離均方差
測站一	測站四	0.004	0.254	0.135
測站一	測站二	0.003	0.242	0.129
測站二	測站四	0.004	0.280	0.152
測站二	測站一	0.003	0.242	0.129
測站四	測站一	0.004	0.254	0.135
測站四	測站二	0.004	0.280	0.152



圖七 整體疊合成果圖

附錄五 竹東整治地控制點資料

表一 竹東芎林測試區統計資料（單位：m）

	X 座標	Y 座標	Z 座標
最小值	257390.55	2734794.52	198.65
最大值	257580.42	2734944.75	262.05
總掃描點數	5163190 點		
面積	28161 m^2		
點密度	183.35 pts/m^2		

表二 控制點座標（**TWD97** 座標系統）（單位：m）

點　位	控制點座標(N,E,H)
S01	(2734914.733, 257495.011, 210.550)
S02	(2734898.348, 257511.170, 219.490)
S03	(2734892.872, 257444.990, 210.414)
S04	(2734879.718, 257468.202, 220.370)
S05	(2734864.743, 257430.374, 227.547)
S06	(2734851.267, 257402.937, 221.439)

圖一 現場控制點佈置位置圖

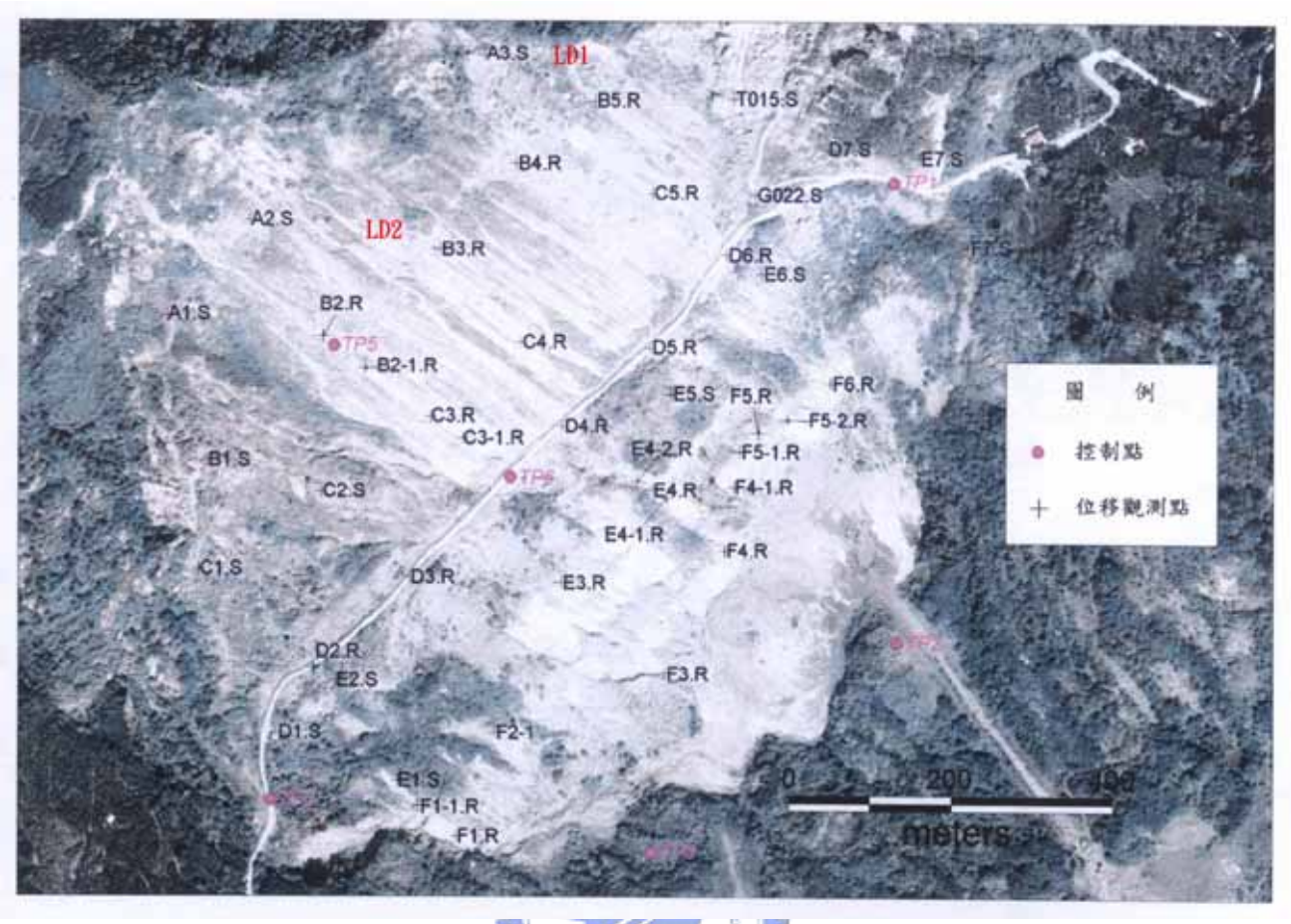
附錄六 九份二山實驗區控制點資料

表一 九份二山實驗區統計資料（單位：m）

	X 座標	Y 座標	Z 座標
最小值	233509.92	2650008.61	599.24
最大值	234650.35	2650892.57	1039.53
總掃描點數	3001262 點		
面積	1006620 m^2		
點密度	2.98 pts/m^2		

表二 控制點座標（**TWD97** 座標系統）（單位：m）

點　位	覘標與控制點距離	控制點座標(N,E,H)
F5.R	1.361	(2650321.243, 234392.184, 653.190)
F5-1.R	1.452	(2650298.800, 234368.153, 655.469)
F4-1.R	1.514	(2650255.794, 234361.427, 640.584)
F4-2.R	1.296	(2650278.310, 234191.342, 710.899)
F5-2.R	1.430	(2650337.882, 234426.451, 646.757)
LD1	1.420	(2650669.060, 234081.449, 840.734)
LD2	1.375	(2650546.416, 233964.418, 852.758)
C3.R	1.201	(2650340.566, 233989.086, 794.704)
C5.R	1.214	(2650609.585, 234261.605, 763.263)



圖一 控制點位置示意圖

（註：崩塌區同時佈有位移觀測點樁，地表主要為岩盤或崩積土，為方便辨別，若控制點樁位於岩盤上，則在點位編號後加 R（如 F5-1.R），若為崩積土則加 S（如 E5.S）。

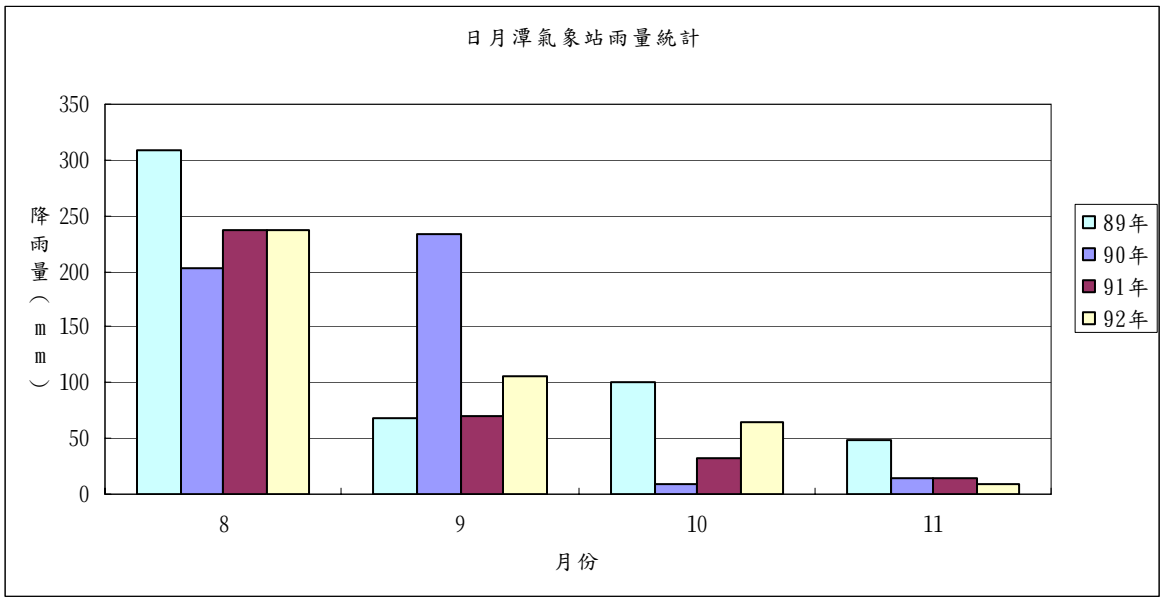
附錄七 九份二山崩塌地滑動監測

本實驗分別於 92/08/27、92/11/25 進行實地掃描作業，紅色圈選部分為主要實驗區範圍，主要目的為應用地面光達技術進行崩塌地監測作業。

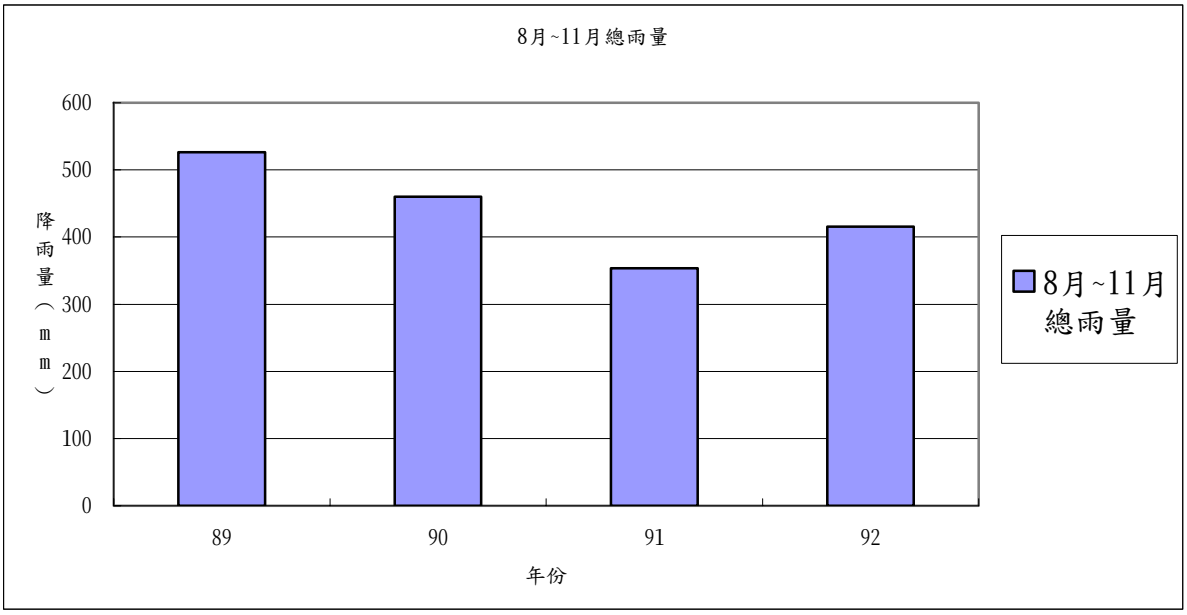
一、測區雨量統計

崩塌之起因與氣候有極大的關係，尤其當颱風或豪雨過後，往往會引起土石流、地滑等天然災害。

距離測區最近的雨量站，為中央氣象局的日月潭雨量觀測站，分析兩次不同觀測日期間（92/08/27~92/11/25）的降雨量，與往年作比較，作為研判是否會造成大量崩塌之原因。



圖一　民國 89~92 年日月潭氣象站降雨量統計圖



圖二　89 年~92 月總降雨量比較圖

二、實驗區數據說明

本次掃描作業分別在 92/08/27 及 92/11/25 兩次不同時間進行觀測，用以分析不同時間該實驗區是否有崩塌情形發生，以達到崩塌地監測之目的。同時由於 8 月 27 號的觀測範圍較小，依其前後兩次掃描成果之重疊範圍選出其中一小塊區域作為實驗區進行崩塌潛勢之分析。

表三為 92/08/27 原始資料數據，表四為實驗區資料數據。

表五為 92/11/25 原始資料數據，表六為實驗區資料數據。

表三　92 年 8 月 27 號原始資料數據（單位：m）

	X 座標	Y 座標	Z 座標
最小值	233890.54	2650151.90	597.52
最大值	234598.61	2650680.72	862.88
總掃描點數	1330327 點		
面積	374003m^2		
點密度	3.56 pts/m^2		

表四　92 年 8 月 27 號實驗區數據（單位：m）

	X 座標	Y 座標	Z 座標
最小值	234200	2650153	597.74
最大值	234450	2650400	742.93
總掃描點數	1290817 點		
面積	61750m^2		
點密度	20.90 pts/m^2		

表五　92 年 11 月 25 號原始資料數據（單位：m）

	X 座標	Y 座標	Z 座標
最小值	233509.92	2650008.61	599.24
最大值	234650.35	2650892.57	1039.53
總掃描點數	3001262 點		
面積	1006620 m^2		
點密度	2.98 pts/m^2		

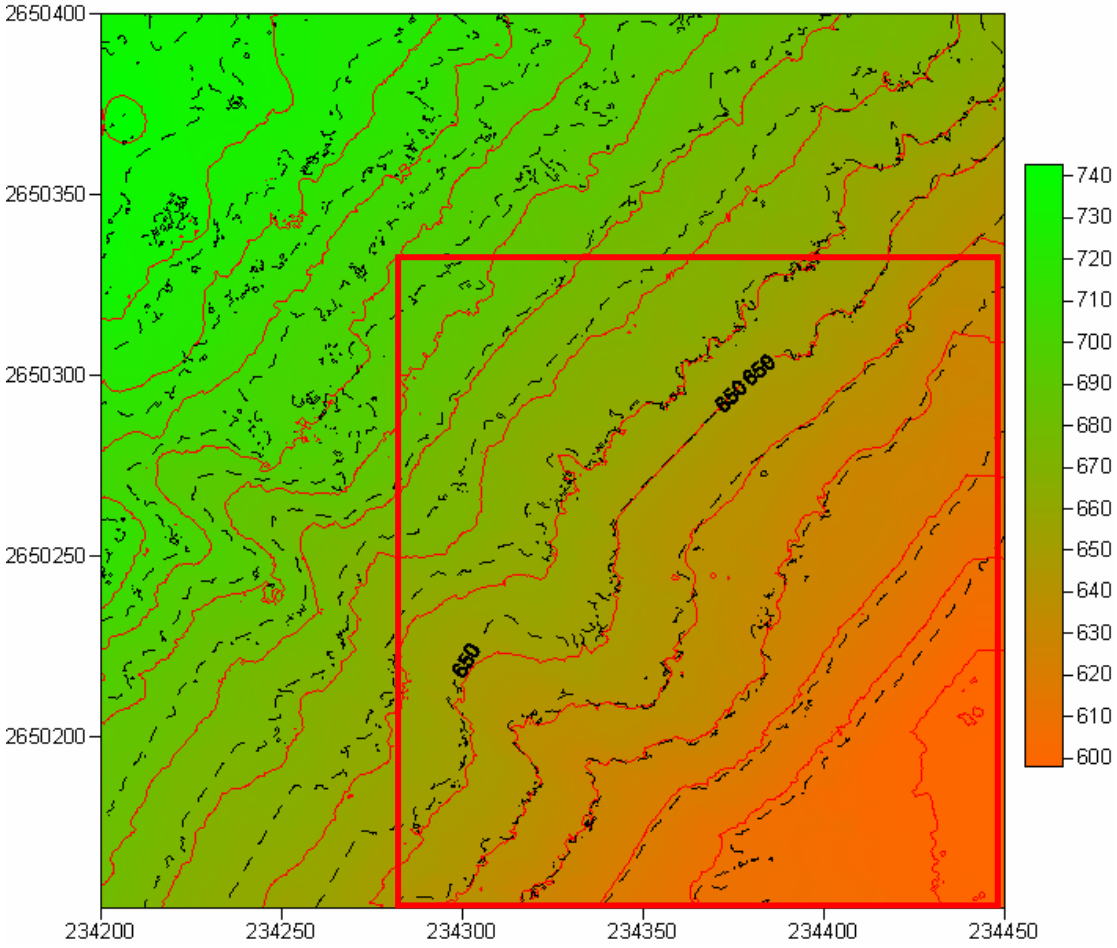
表六　92 年 11 月 25 號之實驗區數據（單位：m）

	X 座標	Y 座標	Z 座標
最小值	234200	2650153	600.46
最大值	234450	2650400	740.66
總掃描點數	876314 點		
面積	61750 m^2		
點密度	14.19 pts/m^2		

三、分析成果比較

a.等值線比較及土石方變化量

下圖三為實驗區中兩個不同時期之等值線差異圖，紅色部分為 92/08/27 的等值線，黑色為 92/11/25 的等值線，紅色圈選處可明顯看到前後兩個不同時間之等值線差異並不大，表七為前後兩次不同時間之土石方變化量。



圖三　8 月 27 號與 11 月 25 號等值線之變化比較圖

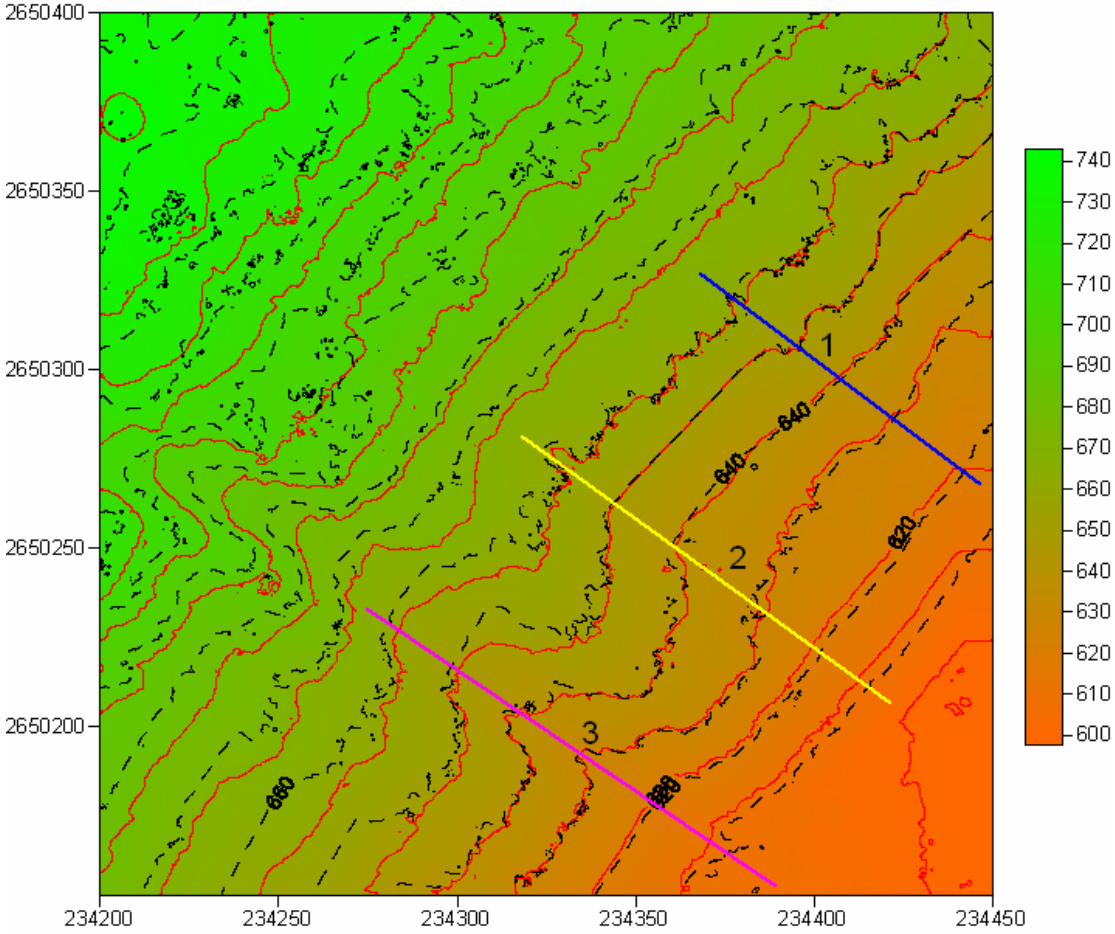
表 7 實驗區土石方變化量

挖方	146439.4309
填方	20291.5425
總體積	126147.8883
面積	61750

單位：m^2

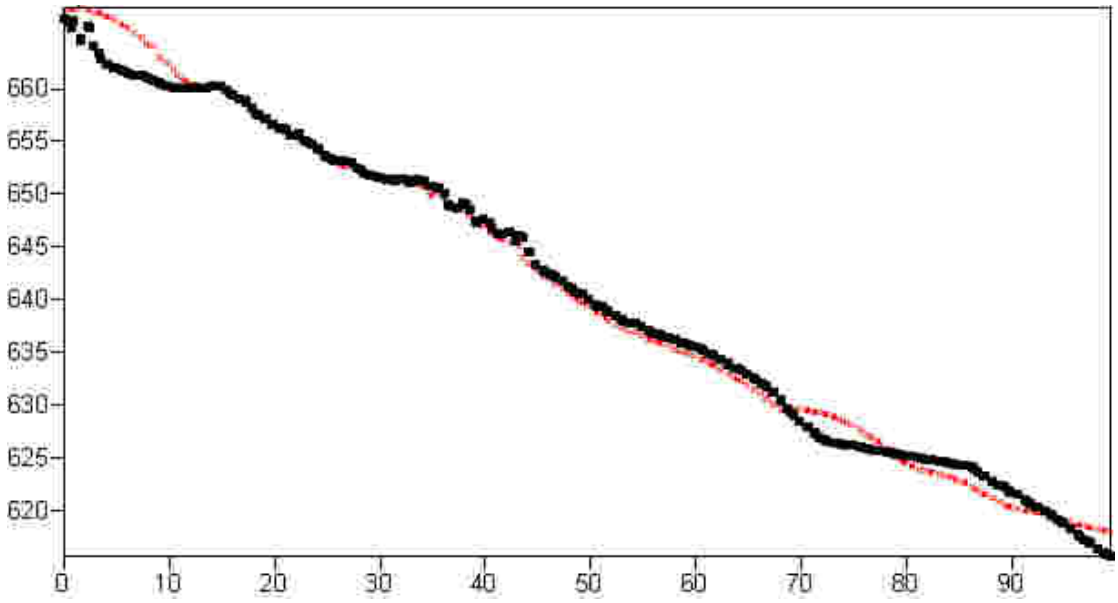
b.剖面圖分析

在圖四中選取三條剖線，藉以分析前後兩次不同時間段，實驗區的高程起伏變化情形，剖線分析圖如圖五～圖七所示，其中紅線（較細）代表 8/27 之資料，黑線（較粗）為 11/25 的資料。



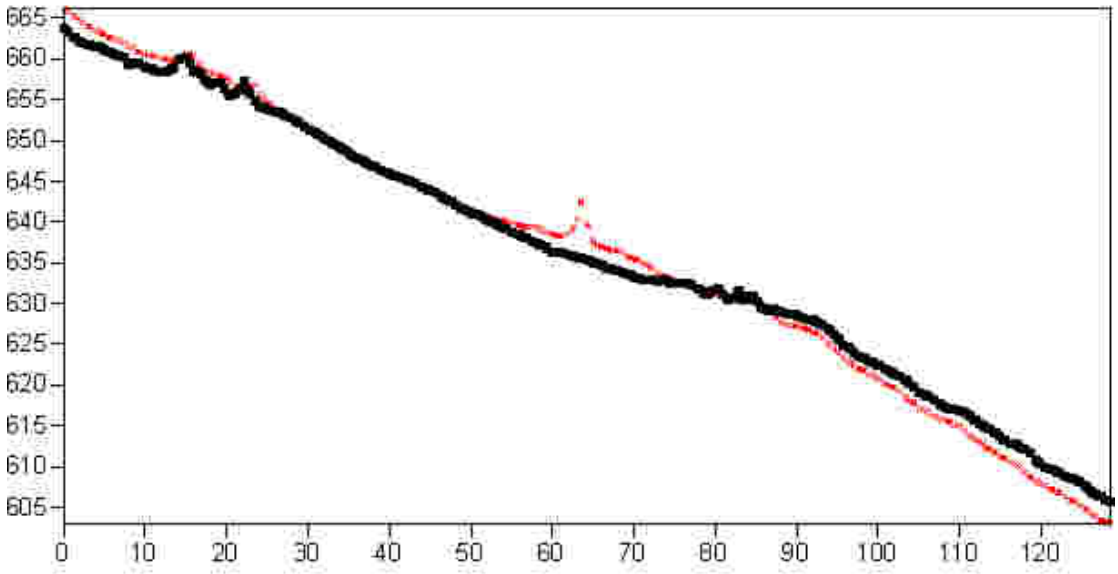
圖四　剖線位置圖

剖線 1 起點座標（234367.71, 2650327.46）→終點座標（234446.97,2650267.76）



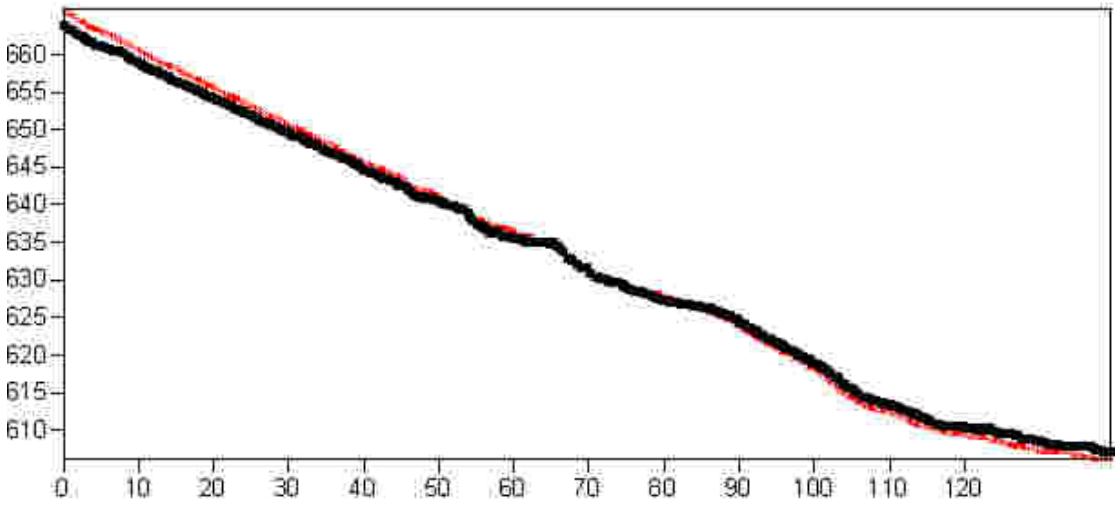
圖五 剖線圖 1

剖線 2 起點座標（234317.56, 2650281.84）→終點座標（234421.4, 2650206.30）



圖六 剖線圖 2

剖線 3 起點座標(234274.53, 2650233.47)→終點座標(234389.70, 2650155.02)



圖七 剖線圖 3

四、小結

1. 地面光達具有快速獲得地表面三維資訊的特點，可在崩塌發生後，即時地取得崩塌地的地表面資訊，並進行分析崩塌前後的高程變化；此外，透過特定的軟體，可達到視覺化之目的，展現觀測地區的三維型態。
2. 由成果分析可知，兩次不同時間的崩塌潛勢並不明顯，這與工研院能資所九份二山崩塌地觀測計畫(2003)的成果報告書相同，工研院利用 GPS 及 EDM 進行位移觀測點樁的監測，其觀測樁移位情形並不明顯。
3. 傳統的崩塌地監測方式是以 GPS 觀測所得之控制點座標做為基準，利用導線測量之方式，觀測位移樁位在不同時間的變化情形，所得到之資訊僅為點的資訊，況且多佈設一個位移樁，就會增加經費的支出；而利用地面光達可獲得面的資訊，透過不同時間序列的掃描作業，可瞭解整個崩塌地的移動潛勢，而達到崩塌地監測之目的。

作者簡歷

姓名：郭朗哲

籍貫：高雄市

出生日期：68 年 1 月 25 日

學歷：高雄市立高雄中學畢業

國立成功大學測量工程學系畢業

國立交通大學土木工程學系肄業