淡江大學機械與機電工程學系碩士班

碩士論文

指導教授：王銀添 博士

使用 RGB-D 感測器實現機器人同時定位、建圖與移動中建立場景

Robot Simultaneous Localization, Mapping and Structure From Motion Using a RGB-D Sensor

研究生：楊仕謙　撰

中華民國 102 年 06 月

誌謝

在這兩年的碩士生涯期間說長不長說短不短，首先，要感謝的是機電系所有師長的指導與栽培，最主要就是我的指導教授王銀添博士，在研究方面上不厭其煩的細心教導我，使得我能完成此篇研究論文。此外也感謝為我評審論文的劉昭華老師、楊子毅老師，以及與指導教授共同合作計畫的孫崇訓老師都給於我的論文許多的建議與指導，以及系上助理小辛、虹吟、惟心助教，與單姊、小金姊提供的多樣資源。

其次，我要感謝與我同一個實驗室的周欣叡大大、巴特大大、黃逸展、蕭大、王晨儒大大、蔡承恩大大、潘致中大大、炫汶大大，忍受我時常不當的言行舉止，以及鄭紹庭、陳庭瑋、袁伯維、劉承霖、涂逸宏、王翰森學弟們幫助完成實驗並提供可靠的攻略，還有 Shoei 學姊、雨政學長、國瑋學長、冠瑜學長，給予我研究上的問題的建議與解決方法；G409 實驗室的阿長、嚕咪、Tako、彥甫哥、子揚、百鈞、育堂學長、俊賢學長、阿琨學長、Magic 學長、G 米學長、建成學長、王耀群學弟，讓我串串門子、打電玩、打保齡球、打嘴砲，以及提供實驗資源給我，我幾乎都快成為他們實驗室的一員；K001-A 實驗室的情獸、威仲、嘉齊、家成和何柏緯等學弟妹們，在沒人陪我的情況下的陪我打電玩；G408 實驗室的陳政偉、王勝弘、陳可辰學長和陳建凱等學弟們，讓我借用電腦並在休息時間陪我消遣打電動；G106 實驗室的王永勳、張妮妮、柏凱、johnson、俊廷學長、凱平學長和陳建榕等學弟們，陪我打球、打嘴砲；K002 實驗室的尹瑞蓮、袁嘉瑩、王建嘉、陳韋賢、張小花、王之之、羅婕、田力仁，讓我平時無聊可以有地方休息打屁聊天；G101 實驗室的王婉萱、葛瑞齡、邱柏翔、紀厚任、昕鴻學長，提供實驗上的建議與器材，以及飼養小魚魚的知識，當然還是有打屁聊天；E808 實驗室的蔡仲達、朱哲明、徐瑋廷、賴俊良、呂岳翰學弟，提供一些我不知道或是不會的資訊，也讓我偶爾串串門子；G304 實驗室的游舒凱等同學們，陪我聊

聊電玩資訊。

再來，要感謝樸毅青年團的學長姊弟妹和同屆的夥伴們，陪我度過這些年的淡江生活，尤其是十二肖人的每一個人，咚咚、阿法、水母、大千、蒲、珮京、蔥哥、巨肚、映晴、聲寶、小龜，在後面幫助我和互相加油打氣，還有學校的大哥姊們，彥君姊、乾媽美蘭姊、豐齊姊、力誠大哥哥，大學到研究所的建議教導並幫助我，還有感謝在淡江認識的一堆幫助過我的朋友們，以及從國中認識到現在的朋友們，Q 毛、毓珊、Zoe、芷瑄，感謝你們在我回宜蘭的時間，再忙也能抽出時間陪我。

最後要感謝的還是我老爸和老媽努力的賺錢提供我就學，偶爾還要聽我抱怨，還有我哥不厭其犯的聽我一直問白癡的問題，還有我的阿姨在我遇到問題的時候提供我一些幫助並幫助我解決問題，最最最最感謝的當然還是我親愛的女友小嫻一直陪著我、幫助我、忍受我，讓我沒陷在困境裡，最愛她了。太多要謝的人了，那些沒有寫到名字的朋友們在這邊先說聲抱歉了，我已經盡力想了，當然不免俗的要說一下名言「太多人要謝了，那就謝天吧!!」，感謝那些曾經直接或間接幫助過我的人，讓我度過這幾年的研究生活和完成我的學業，也住這些人也一樣事事順心!

楊仕謙 謹誌

102 年 7 月於 E806 實驗室

論文名稱：使用 RGB-D 感測器實現機器人同時定位、 頁數： 48
建圖與移動中建立場景

校系(所)組別：淡江大學機械與機電工程學系碩士班
畢業時間及提要別： 101 學年度第 2 學期 碩士 學位論文提要
研究生： 楊仕謙 指導教授： 王銀添 博士

論文提要內容：

本論文使用微軟 Kinect 之 RGB-D 感測器，發展機器人同時定位與建圖演算法。研究分為四個階段：第一階段校準 Kinect RGB-D 感測器，包括 RGB 攝影機的內部參數校準，以及 RGB 鏡頭與深度鏡頭的歪斜校準。基於 RGB-D 感測器的同時定位與建圖之演算法在第二階段被建立與實測。第三階段將移動中建立場景的功能整合到同時定位與建圖任務中，以便建立環境模型。第四階段規劃系統的運算速度之改善程序。利用雲端運算的概念，將運算系統分為影像處理與狀態估測兩個運算程序。將影像處理程序保留在移動感測系統端，而狀態估測程序交由雲端運算伺服器進行處理。實測結果顯示使用本論文規劃的雲端運算程序，可以加快 15%的運算速度。

關鍵字：RGB-D 感測器、同時定位與建圖、場景重建、雲端運算

表單編號：ATRX-Q03-001-FM030-01

Title of Thesis: Robot Simultaneous Localization, Mapping and Structure From Motion Using a RGB-D Sensor **Total pages:** 48

Key word: RGB-D Sensor;SLAM;SfM;Cloud Computing

Name of Institute: Department of Mechanical and Electro-Mechanical Engineering, Tamkang University.

Graduate date: June, 2013 **Degree conferred: Master**

Name of student: Shih-Chien Yang **Advisor: Yin-Tien Wang**

楊 仕 謙 **王 銀 添**

Abstract:

This thesis presents an algorithm of robot simultaneous localization and mapping (SLAM) using a RGB-D sensor. This research consists of four stages: first, the Kinect RGB-D sensor is calibrated including the intrinsic parameters of RGB camera as well as the alignment of the RGB sensor and the depth sensor. The RGB-D SLAM is developed and implemented in indoor environments at the second stage. Third, the task of structure from motion (SFM) is integrated with the RGB-D SLAM to construct the environment model. Computational speed is improved at the last stage. The concept cloud computing is applied to the SLAM system by dividing the system into two procedures including image processing and state estimation. The procedure of image processing is remained at the mobile sensory system, while the state estimation is implemented by a cloud computing server. Experimental results show that the computational speed is increased 15% with the cloud computing.

表單編號：ATRX-Q03-001-FM031-01

目錄

圖目錄

表目錄

第1章 序論

1.1 研究動機與目的

本研究使用微軟 Kinect 感測器發展機器人同時定位與建圖(simultaneous localization and mapping, SLAM)演算法，以便在未知環境中執行巡航任務，以及將巡航過的場景利用存下來的數據，使用場景重建的方法達到地圖的場景重建。並且運用雲端計算的概念將原本複雜的運算程序交由雲端伺服器計算，以便分擔計算的負擔及增加運算的速度。

1.2 文獻探討

1.2.1 Kinect 感測器影像校準

由於使用 C++撰寫 Kinect 感測器應用程式的資料非常少，所以參考學者[1]探討彩色與深度資料的格式文章以及微軟官方[2]的原始碼加以了解。在感測器校準的領域，已有許多學者[3][4][5][6]探討影像校準的研究，本論文參考這些論文將影像的扭曲和歪斜校準，運用在 Kinect 的影像校準上。

1.2.2 Kinect EKF SLAM

在本論文是使用馮盈捷[7]SLAM 的程式並加以改寫，並使用 Kinect 感測器實現機器人同時定位與建圖的巡航任務。

1.2.3 場景重建

在場景重現有許多論文的探討[8][9]，但方法有很多種，本論文研究以論文[9]加以修改並呈現場景重建。

1.2.4 雲端計算

雲端計算的文章有[10]，但由於設備上的限制，本論文先以檔案傳輸協定(File Transfer Protocol,FTP)[11]的方式加以實現本論文的雲端計算功能。

1.3 研究範圍

1.3.1 深度資料的建立

了解微軟 Kinect 感測器的深度資料格式，並加以解釋，使 Kinect 感測器所擷取到的深度可以運用在擴張型卡爾曼濾波器(extended Kalman filter, EKF) SLAM 的演算法上，來達到機器人巡航之任務。

1.3.2 Kinect 感測器校準

校準微軟 Kinect 感測器的 RGB (red-green-blue)感測器的內部參數，以及 RGB 感測器與紅外線深度感測器在影像平面的歪斜參數之校準。RGB 感測器的內部參數方面將使用[5]的校準方法，而影像平面歪斜參數[6]的校準則使用非線性多項式模擬歪斜現象。

1.3.3 場景重建

場景重建將規劃行動中建立結構(structure from motion, SFM)的功能，而 SFM 功能將依附在機器人 SLAM 的任務中實現。將使用 EKF 做為機器人 SLAM 的估測器。

1.3.4 雲端計算

EKF SLAM 的預測與更新運算模組相當費時。本論文將使用雲端計算的概念，將費時的運算模組分配到另一台桌上型電腦上，並利用網路傳輸資料將資料送給桌上型電腦做運算，再將結果回傳給機器人。而影像處理則使用機器人的機上電腦進行運算，靠近影像擷取端以便節省影像傳輸的成本。

1.4 系統架構

本論文使用的工具與技術包括以雙眼視覺(binocular vision)做為感測模組，使用加速強健特徵(Speeded-Up Robust Features, SURF)[12]建立環境建圖，利用擴張型卡爾曼濾波器進行 SLAM[13]等。Kinect 感測模組的移動方式包括使用手持(hand-holding)方式移動，裝設在差速驅動行動機器人系統上跟隨著機器人移動，以及裝設在飛行機器人從空中或高處進行 SLAM。

1.5 論文架構

本論文共分為六章:第二章將以 Kinect 感測器為架構，剛開始說明感測器的系統以及感測器的校準，主要探討深度資料格式以及校準的問題，包含校準出來的參數並對其量化做比較。第三章主要為實現Kinect EKF SLAM並做兩個範例。第四章則是建立場景重建之系統，並對其演算法做探討，並做其範例。第五章為雲端運算，並利用 FTP 將 EKF SLAM 演算法分成兩台電腦做運算加以實驗並量化比較。第六章為本論文研究成果與未來研究的方向並針對本論文提出具體貢獻。

第2章 Kinect 感測器的校準

應用如圖 2.1 所示的 Kinect 感測器在視覺辨識中，必須以影像處理方式得知特徵點。但是在尋找影像特徵點的過程中，Kinect 感測器本身的硬體上的關係而產生許多的誤差問題，其中最為重要的是 RGB 影像和深度影像的偏移問題。

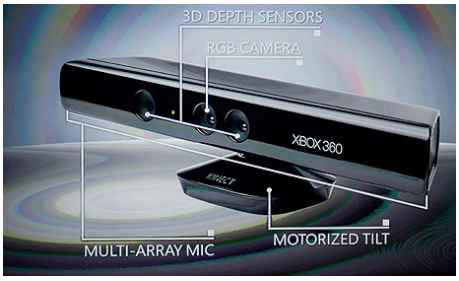


圖 2.1 Kinect 感測器

表 2.1 Kinect 感測器規格表

感應項目	有效範圍
顏色與深度	1.2 ~ 3.6 公尺
骨架追蹤	1.2 ~ 3.6 公尺
視野角度	水平 57 度、垂直 43 度
底座馬達旋轉	左右各 28 度
每秒畫格	30 FPS
深度解析度	QVGA (320 x 240)
顏色解析度	VGA (640 x 480)
聲音格式	16KHz, 16 位元 mono pulse code modulation (PCM)
聲音輸入	四麥克風陣列、24 位元類比數位轉換 (ADC)、雜音消除

表 2.2 NB 規格表

廠牌	ASUS
型號	A43S
CPU	Intel Core i7-2670QM 2.2GHz
OS	Windows 7 Service Pack 1
RAM	DDR3 4GB

2.1 深度資料

深度影像是由紅外線發射器發射紅外線，經物體反射再由紅外線 CMOS 攝影機接收，所紀錄的影像如圖 2.2、圖 2.3 所示。在深度影像資料串流中，使用 16 位元(bits)描述像素座標(x,y)的深度訊息，單位為 mm，深度距離範圍 850mm~4000mm。深度 0 表示未知狀況，例如該位置是影子、過低反射(玻璃)、過高反射(鏡子)。深度資料是一維陣列，從影像的左上角開始，先從左到右，再從上到下。

開啓串流時由 ImageFormat 指定像素格式有兩種方式：無玩家編號與有玩家編號，參數分別為 NUI_INITIALIZE_FLAG_USES_DEPTH 與 NUI_INITIALIZE_FLAG_USES_DEPTH_AND_PLAYER_INDEX。兩種方式都是使用兩個 byte 的 16 位元資料，第一種方式使用 16 位元資料，第二種方式保留最低 3 位元為玩家編號(0~7)，0 代表沒有玩家，1 代表骨架 1，以此類推，如圖 2.4。例如要計算深度影像左上角像素座標(0,0)所代表的距離，第一種方式從記憶體中複製深度訊息的指令為

memcpy(m_depthD16,LockedRect.pBits,LockedRect.size) (2.1)

其中 m_depthD16[index]為所定義的距離變數；LockedRect.pBits 為 Kinect SDK 鎖住的方形記憶體；LockedRect.size 為方形記憶體的大小。第二種方式從記憶體中複製深度訊息的指令為

memcpy(m_depthD16,LockedRect.pBits,LockedRect.size)

距離=(m_depthD16[index] & 0xFFF8)>>3 (2.2)

單位：1mm

本論文使用第二種方式指定格式。

圖 2.2 RGB 影像

圖 2.3 深度影像

2.2 Kinect 鏡頭歪斜問題

Kinect有兩個鏡頭，但兩個鏡頭會因為自動化封裝關係，會有傾斜、偏移或旋轉某一個角度，這些因素都會造成誤差與兩影像對到特徵點不同的位置的情況發生，如圖 2.2、圖 2.3。對於此誤差的修正，可以使用影像幾何轉換方式來處理[6]。以一個四邊形區域的幾何失真轉換為例，如圖 2.5為失真的與對應的校準影像中的四邊形區域，四邊形的頂點是對應的連接點。利用一對雙線性方程式為模型，如方程式(2.3)、(2.4)，因為總共有8個已知的連接點，所以由這些程式可以解出八個係數 $c_i, i=1,2,\ldots,8$。這些係數構成了幾何失真模型，該模型用來把連接點所定義的四邊形區域內的所有像素加以轉換。所以只要在實驗前把影像中場地的左上、左下、右上和右下四個頂點與正確的座標做映射就可以求出符合的係數值，將來只要輸入影像中目標物的座標代入方程式(2.3)、(2.4)，就可以得到正確的目標物座標，修正因為鏡頭歪斜造成的誤差。

$$x''' = r(x,y) = c_1 x + c_2 y + c_3 xy + c_4 \tag{2.3}$$

$$y''' = s(x,y) = c_5 x + c_6 y + c_7 xy + c_8 \tag{2.4}$$

由於深度影像是經過計算後的影像，會產生誤差或殘影如圖 2.7，這樣就無法很精準地找出每點的像素值，這樣會使校準增加另一種誤差，所以本論文要擷取未運算過的紅外線攝影機的影像去做校準，如圖 2.8。

取得IR影像後，利用RGB影像與IR影像如圖 2.9、圖 2.10，各取棋盤格上20個像素點，並代入方程式(2.3)、(2.4)，因為有已知的連接點，所以由這些程式可以解出八個係數 $c_i, i=1,2,\ldots,8$，如表 2.3。

表 2.3 歪斜校準參數

	C1	C2	C3	C4	C5	C6	C7	C8
數值	1.0878	-0.0100	0.0001	-27.4165	-0.0029	1.0970	-0.0000	-21.2797

得到 8 個係數構成的幾何失真模型後，該模型用來把所定義的影像區域內的所有像素加以轉換。所以只要在做實驗前將 RGB 影像與 IR 影像的座標做映射就可以求出符合的係數值，將來只要得到 RGB 影像中的座標代入方程式(2.3)、(2.4)就可以得到深度影像的座標，修正因為計算造成的誤差。把這些得到的係數代回方程式並對每個像素做歪斜校準，以 Matlab 呈現如圖 2.11、圖 2.12、圖 2.13，很明顯的 RGB 影像與校準後的深度影像，之後再以校準前與校準後的各挑幾個相同角點的像素與 RGB 影像像素，利用誤差距離方程式(2.5)，並劃出直方圖，如圖 2.17，明顯看的出來校準後的像素與 RGB 影像的像素較為相近。

$$\text{誤差距離} = \sqrt{(Ix - Ix')^2 + (Iy - Iy')^2} \qquad (2.5)$$

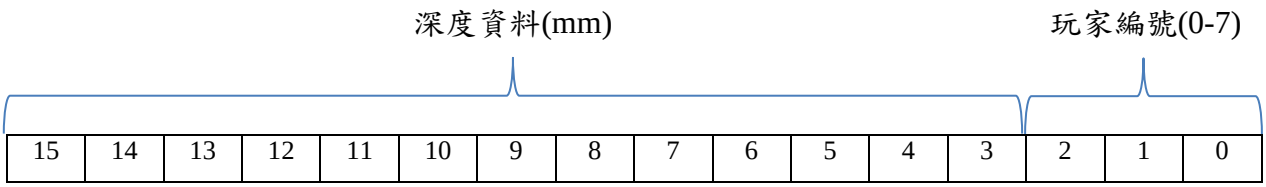


圖 2.4 深度格式

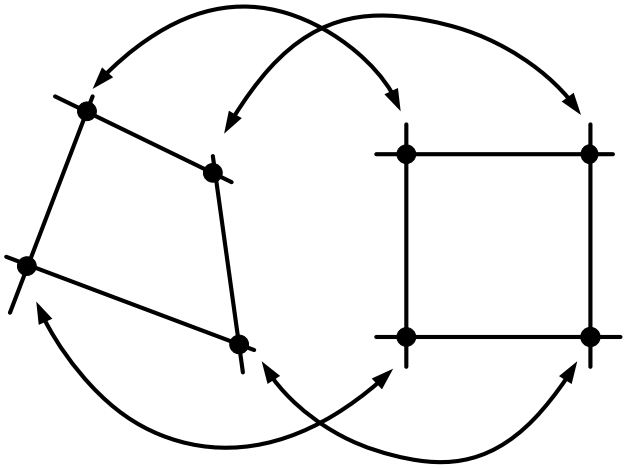


圖 2.5 兩個影像片段中對應的點

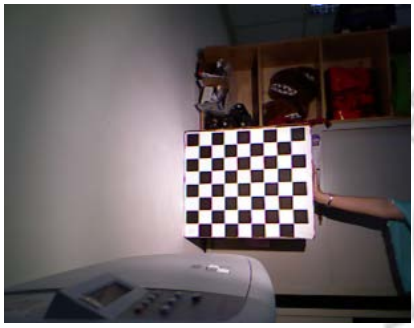


圖 2.6 棋盤格的 RGB 影像

圖 2.7 棋盤格的深度影像

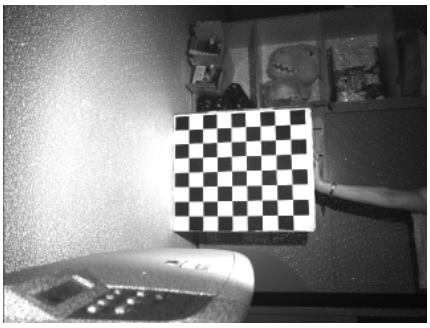


圖 2.8 棋盤格的 IR 影像

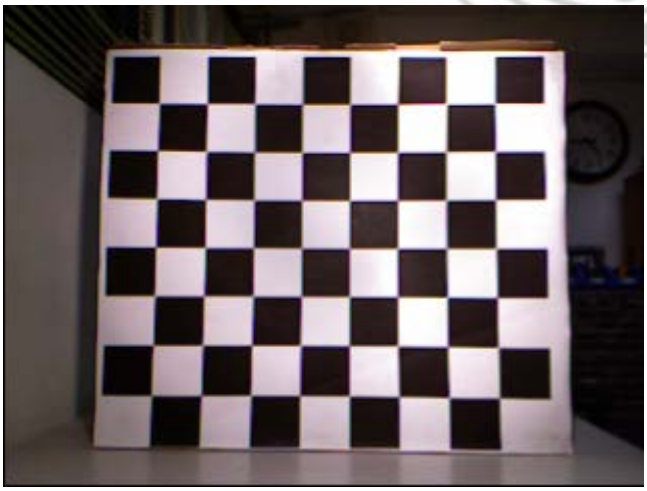


圖 2.9 校準用 RGB 影像

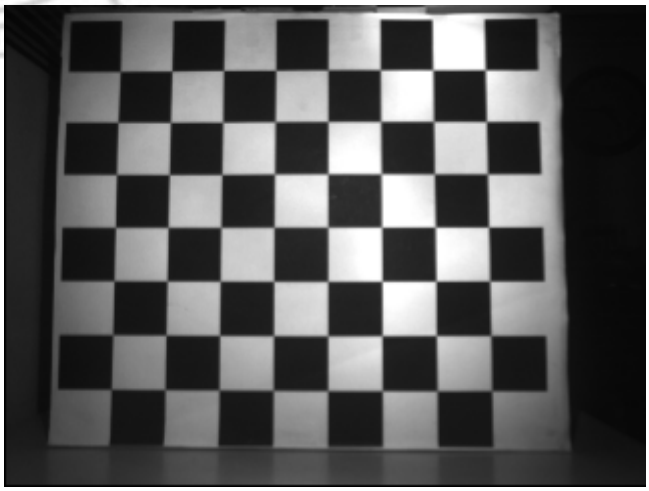


圖 2.10 校準用 IR 影像

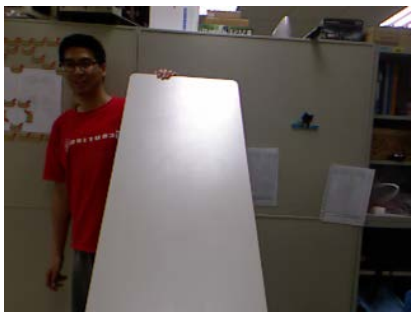


圖 2.11 RGB 影像

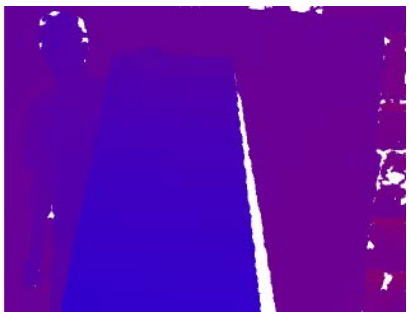


圖 2.12 未校準深度影像

圖 2.13 校準後深度影像

圖 2.14 RGB 棋盤格影像角點

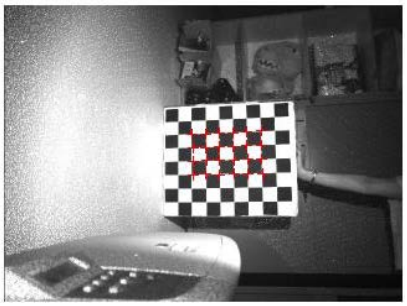


圖 2.15 IR 棋盤格影像角點

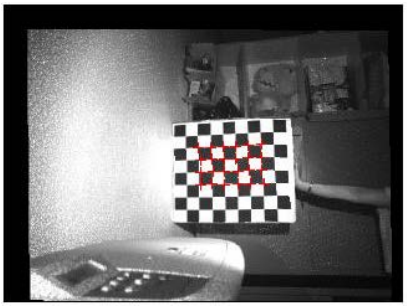


圖 2.16 校準後 IR 棋盤格影像角點

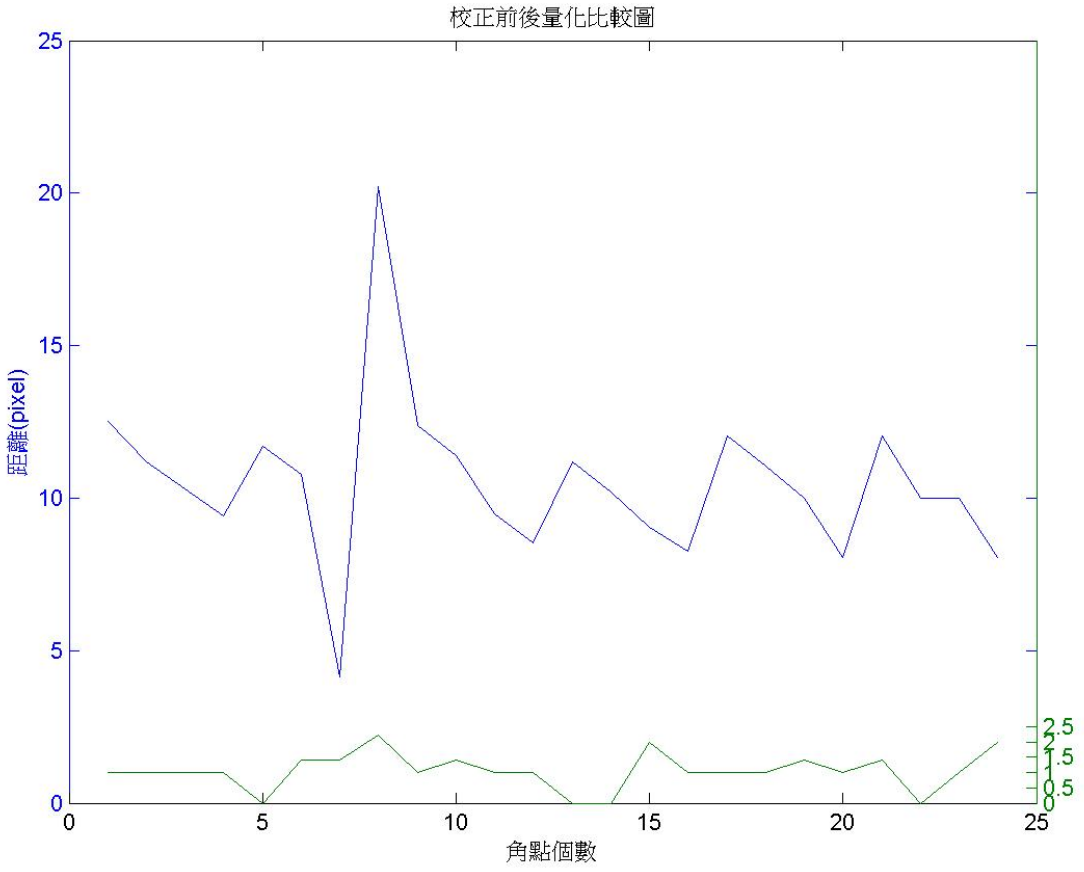


圖 2.17 校準影像像素與 RGB 影像像素誤差距離比較直方圖

2.3 Kinect RGB 攝影機校準

這個世界不是完美的，即使你花大錢買了十幾萬、二十幾萬的相機，昂貴的鏡頭照出來的照片和真實的景物還是有一定的誤差，即時那誤差很小，但是絕對是存在的，Kinect 感測器也會有些許的誤差存在，因此我們為了要更精準的像素計算，我們要校準攝影機的內部參數，求解焦距與外部參數[4]、求解鏡頭的扭曲與變形參數[3]、影像修正[5]；步驟如下：拍攝 20 張棋盤格照片(黑白相間)，每張棋盤格照片需用不同的角度去拍攝，但每張旋轉的角度不要超過 90 度，如圖 2.18。讀入 20 張圖，各自對 20 張照片進行操作，按順序點選這 20 張圖的四個角落，如圖 2.19。點選 Calibration，得出結果，如表 2.4、表 2.5 與圖 2.20 的四張圖。

表 2.4 內部參數

	變數名稱	數值(單位)
Focal Length	f_u	260.31000(pixel)
	f_v	260.84923(pixel)
Principal point	u_0	163.75230(pixel)
	v_0	126.43371(pixel)
Distortion	$k_d(1)$	0.23311
	$k_d(2)$	-0.64002
	$k_d(3)$	-0.00528
	$k_d(4)$	0.00357
	$k_d(5)$	0.00000

表 2.5 影像修正模型參數

	$k_c(1)$	$k_c(2)$	$k_c(3)$	$k_c(4)$	$k_c(5)$	$k_c(6)$	$k_c(7)$	$k_c(8)$
數值	-0.2397	0.6575	0.0053	-0.0037	0.2731	-0.7065	-0.0040	0.0045

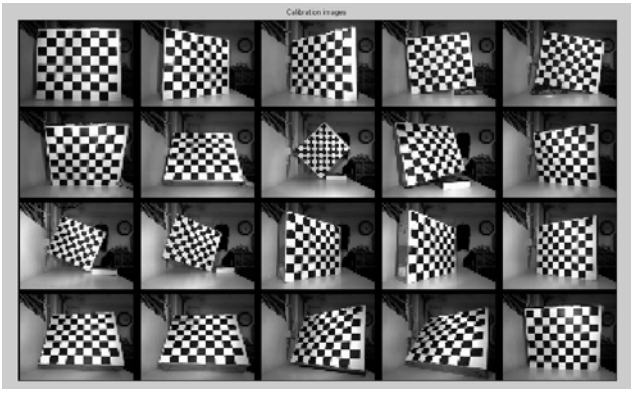


圖 2.18 二十張棋盤格照片

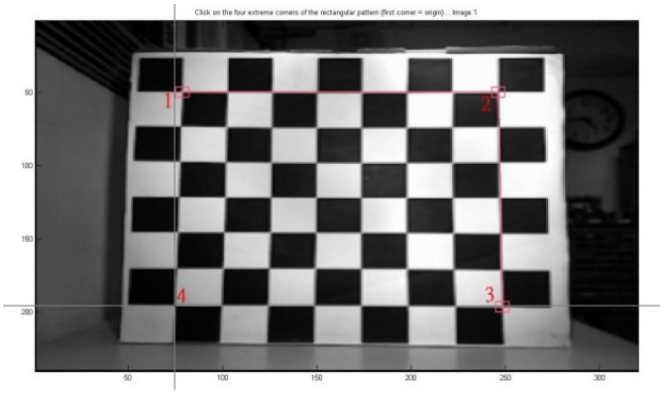


圖 2.19 依序將每一張圖都點選角落

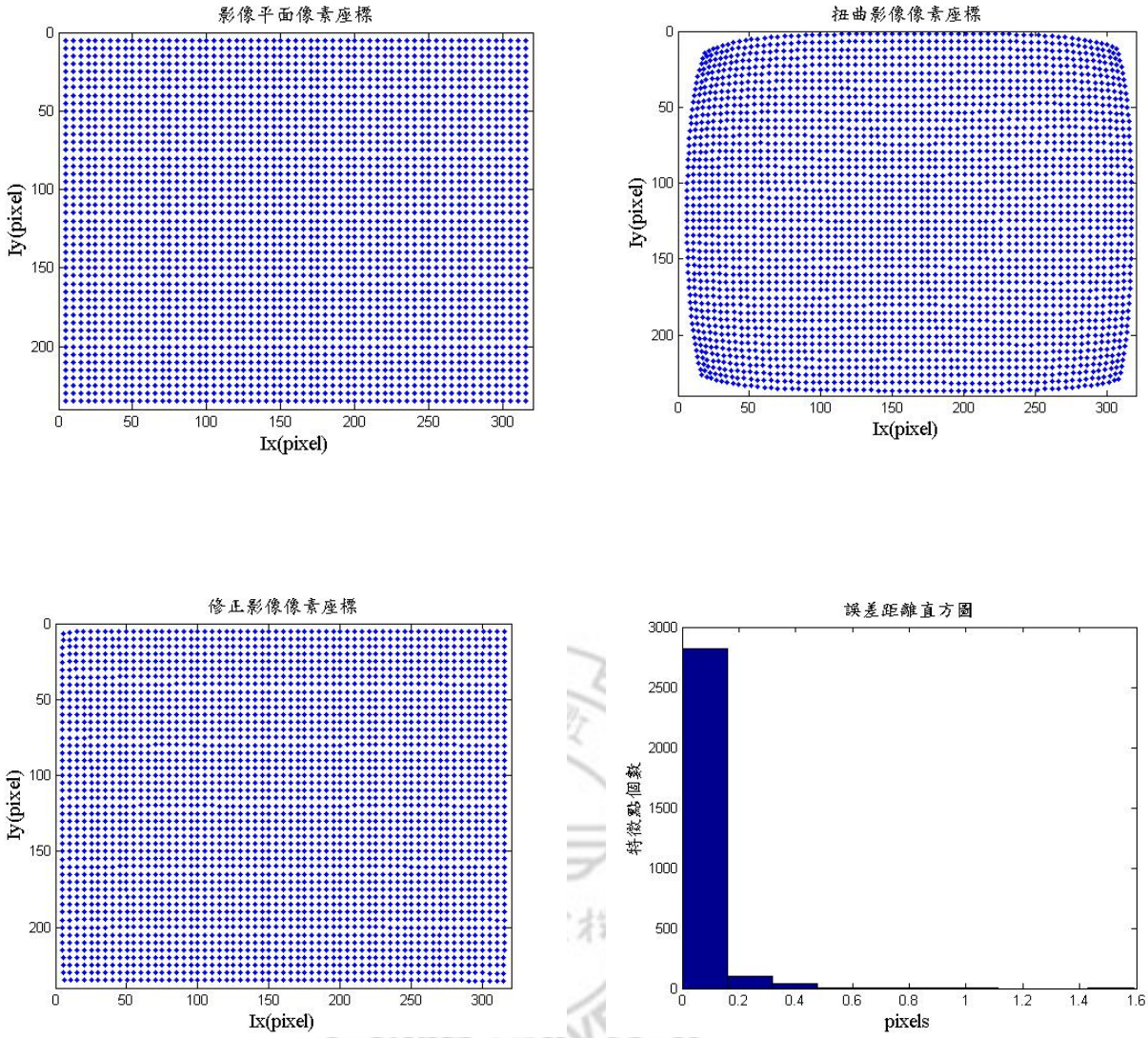


圖 2.20 matlab 計算結果之像素座標圖

第3章 Kinect EKF SLAM

本章使用馮盈捷[7]所建立的雙眼視覺 EKF SLAM 系統進行範例的實測。所使用之 Kinect EKF SLAM 系統，如圖 2.1 所示，由 Kinect 感測器建立而成，其規格如表 2.1 所示。使用之筆記型電腦規格如表 2.2 所示。Kinect 感測器歪斜校正參數使用 Gonzalez R.C., and R.E. Woods[6] 影像幾何轉換方式來求算，取得 Kinect 感測器歪斜校正參數，如表 2.3 所示。Kinect 感測器內部參數使用 Bouguet[3]所發展的攝影機校正程式求算，取得 Kinect 感測器的內部參數，如表 2.4 所示。影像修正模型參考邱明璋[14]所規劃的程序，並求算 Kinect 感測器的影像修正參數，如表 2.5 所示。

3.1 Kinect 感測器同時定位與建圖(SLAM)

此範例將介紹以最小尤基里德距離門檻 d_{match} 作為特徵比對條件，且以 16 維度為影像特徵描述向量，針對左影像上與地圖上所產生的現象加以說明。為了突顯動態搜尋視窗的運作機制，將會在影像上顯示出地標特徵比對成功時所用的搜尋視窗大小。

SLAM 系統解說圖如圖 3.1、圖 3.2，圖 3.1 顯示以下資訊：地標特徵(粗方框)、地標的編號、搜尋視窗(細方框)、偵測此影像所使用的 Dthreshold 值、影像上的地標特徵數、地圖內的地標數、目前系統的狀態(Mapping 或 Lost)，其中 Mapping 為系統執行 SLAM 任務中，而 Lost 為系統無法取得地標量測資訊造成迷航。圖 3.2 以 Matlab 繪製出 Kinect 感測器與地標在地圖 x-y 平面上的位置及其高斯分佈狀態(3 倍標準差)，並顯示 Kinect 感測器所行走過的路徑。本論文以 SLAM 系統啟動位置定義為地圖上世界座標原點，單位為公尺(m)，因此，初始 Kinect 感測器狀態與共變異數均設定為零。其他參數設定值為：

特徵偵測音階數：Octaves = 2

特徵偵測各音階的層數：Octaves Layers = 2

地標特徵新增間隔視窗：Wa = 51×51(pixels)

地標特徵相似門檻值：dsimilar = 0.3

特徵比對最小搜尋視窗：Ws= 15×15(pixels)

特徵比對搜尋視窗單位增加值：10(pixels)

地標特徵最大距離值：maximum distance=4000(mm)

地標特徵最小距離值：minimum distance=850(mm)

地標特徵資料關聯測試限制次數：Nerase = 10

地標特徵資料關聯成功率門檻：Rerase = 0.7

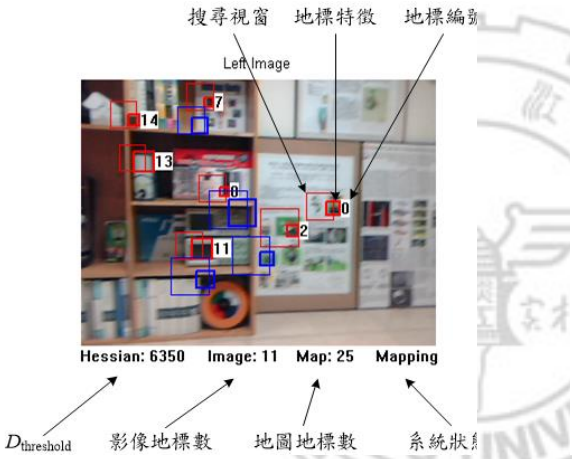


圖 3.1 SLAM 系統狀態

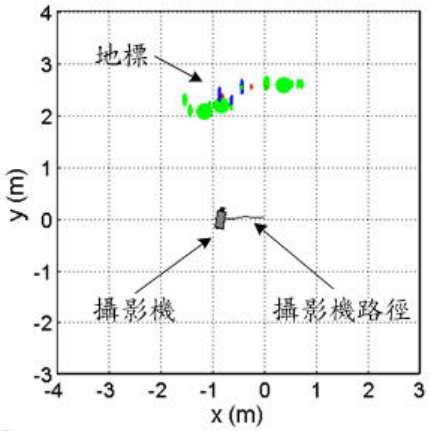


圖 3.2 SLAM 上視圖

3.2 範例一：行走直線路徑

本實驗將 Kinect 感測器直線行走實驗室執行自我定位任務，進行實驗室內部環境概略性建圖，目的是為測試本系統在 Kinect 感測器自由直線移動時的穩定性。以下為偵測路徑和定位的過程之影像擷取：

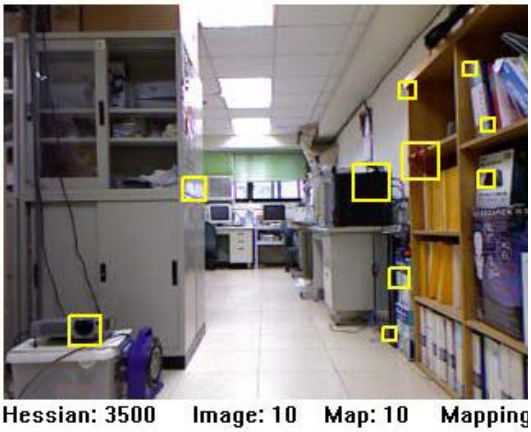


圖 3.3 1st 彩色影像

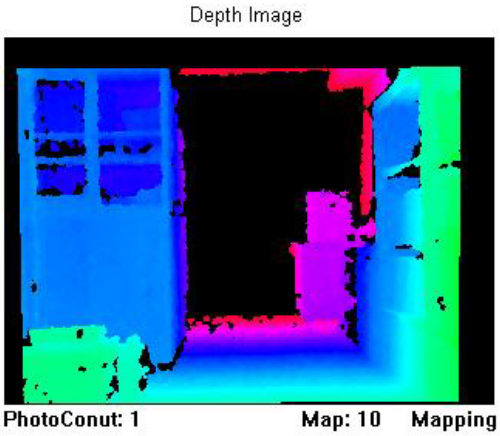


圖 3.4 1st 深度影像

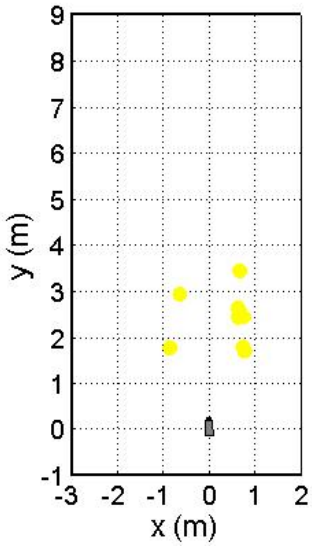


圖 3.5 1st 偵測到新的特徵點

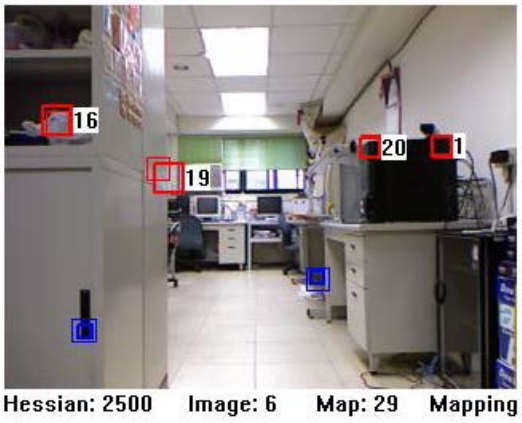


圖 3.6 80th 彩色影像

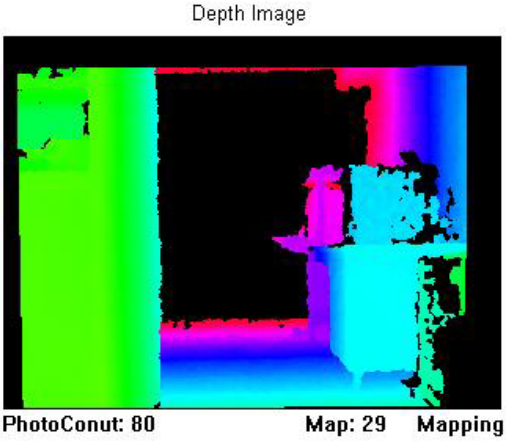


圖 3.7 80th 深度影像

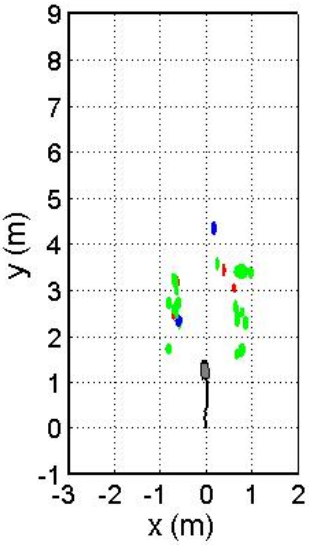


圖 3.8 80th 判斷中與穩定特徵點

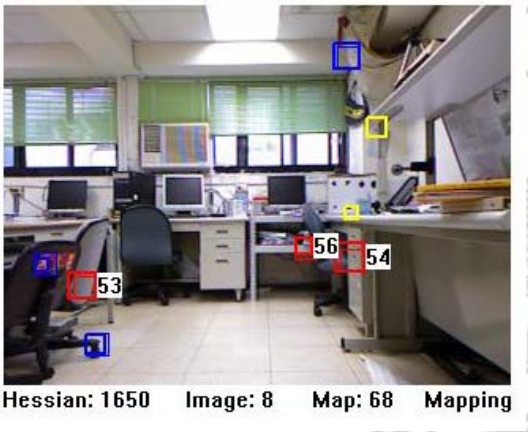


圖 3.9 160th 彩色影像

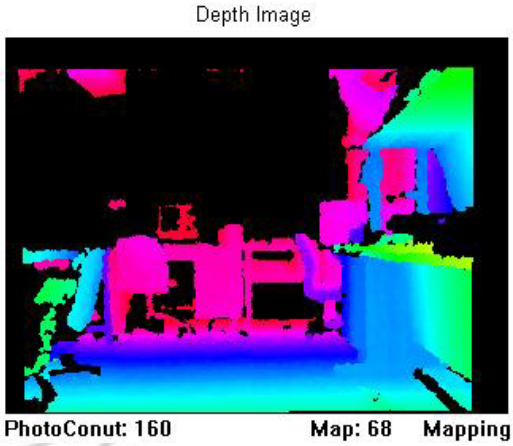


圖 3.10 160th 深度影像

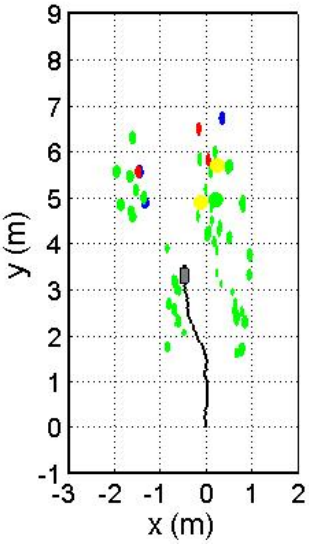


圖 3.11 160th 持續偵測

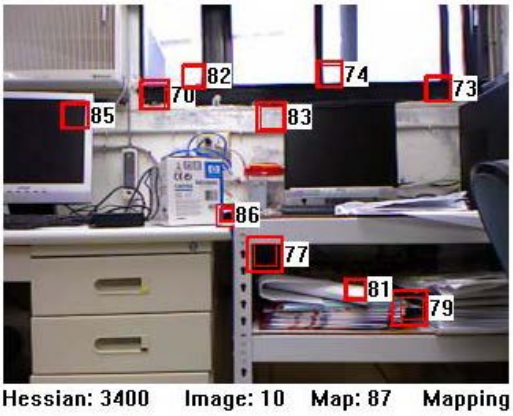


圖 3.12 257th 彩色影像

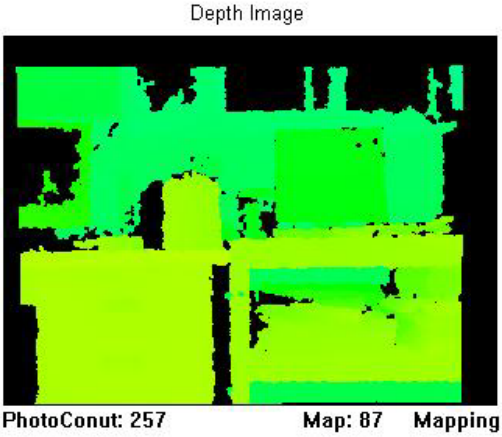


圖 3.13 257th 深度影像

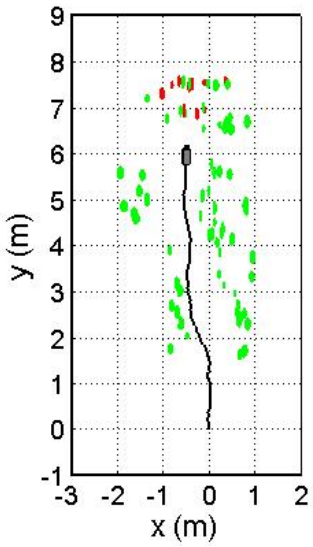


圖 3.14 257th 路徑結束

3.3 範例二：繞行圓形路徑

本實驗將 Kinect 感測器繞行實驗室執行自我定位任務，進行實驗室內部環境概略性建圖，目的是為測試本系統在 Kinect 感測器自由繞行移動時的穩定性。以下為偵測路徑和定位的過程之影像擷取：

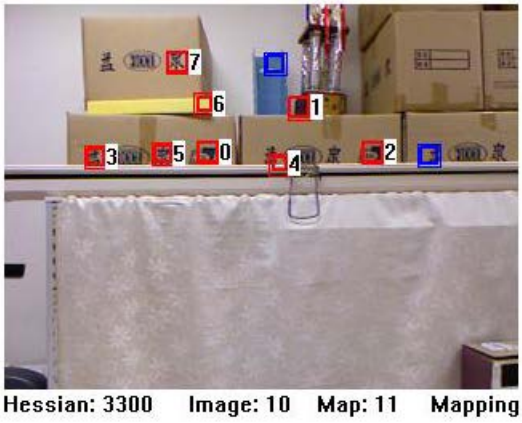


圖 3.15 8th 彩色影像

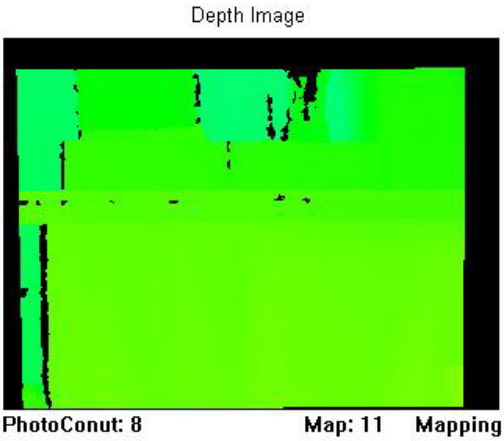


圖 3.16 8th 深度影像

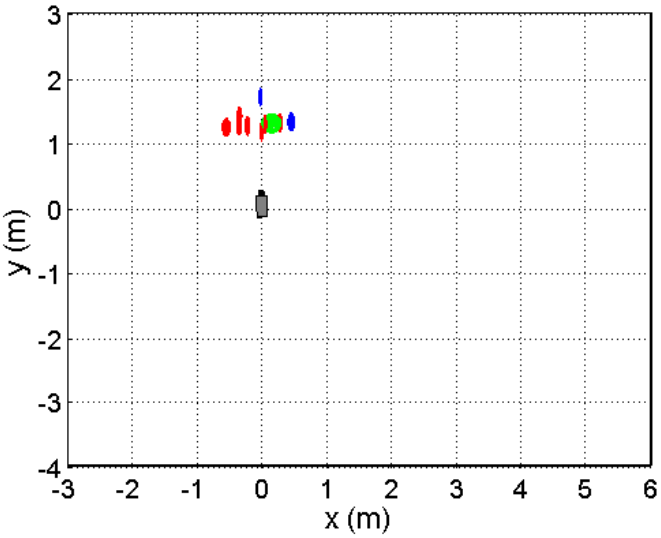


圖 3.17 8th 偵測到穩定特徵點

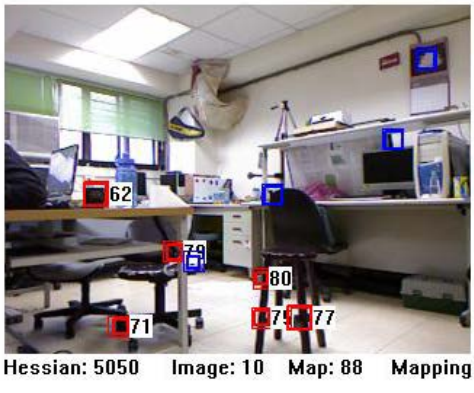


圖 3.18 270th 彩色影像

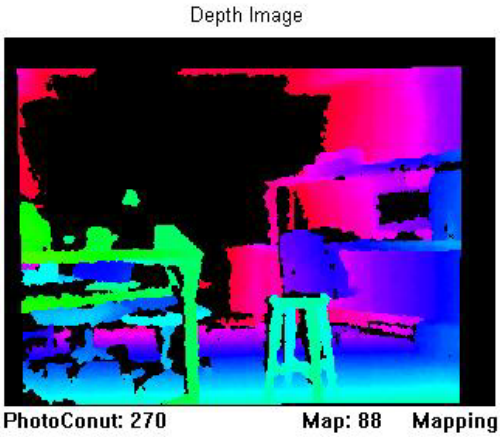


圖 3.19 270th 深度影像

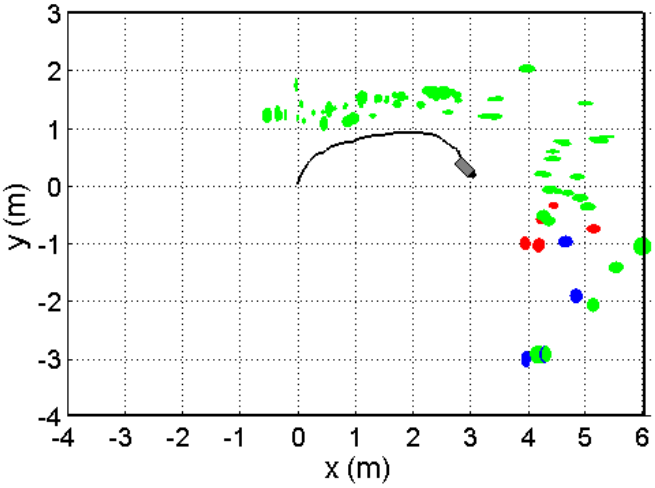


圖 3.20 270th 第一次轉彎

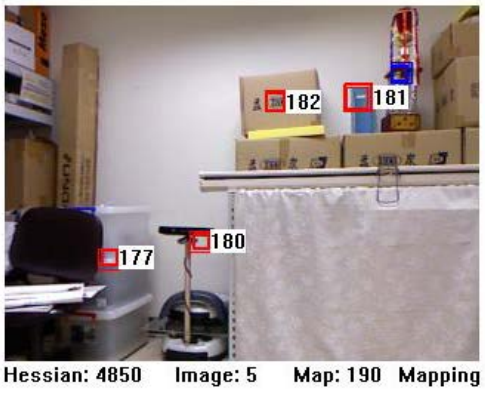


圖 3.21 610th 彩色影像

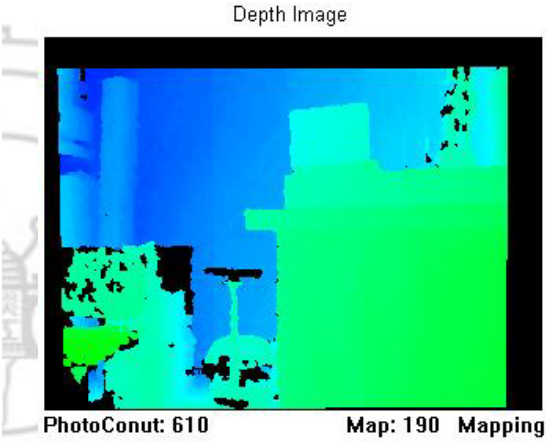


圖 3.22 610th 深度影像

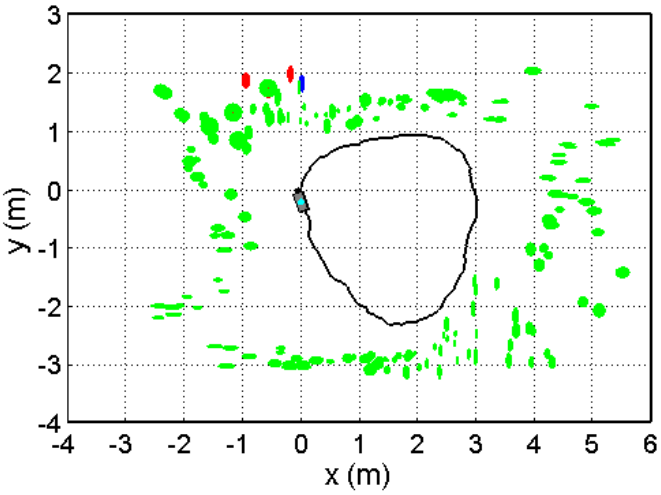


圖 3.23 610th 第一次路徑重合

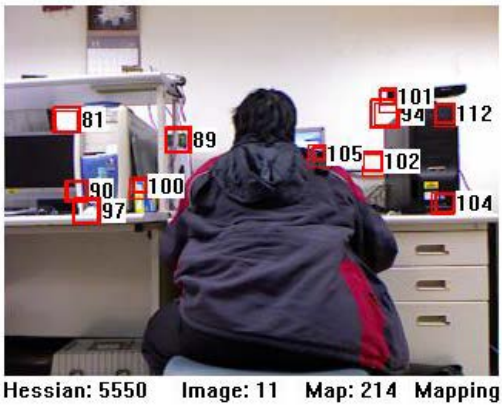


圖 3.24 850th 彩色影像

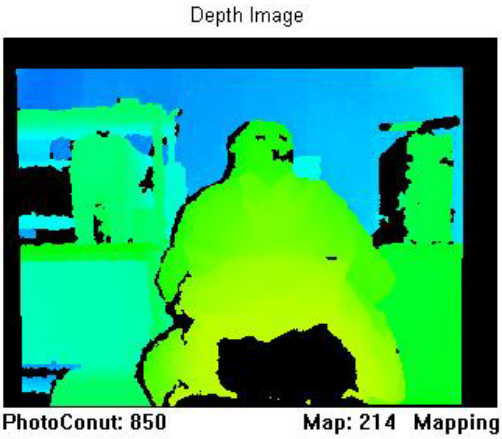


圖 3.25 850th 深度影像

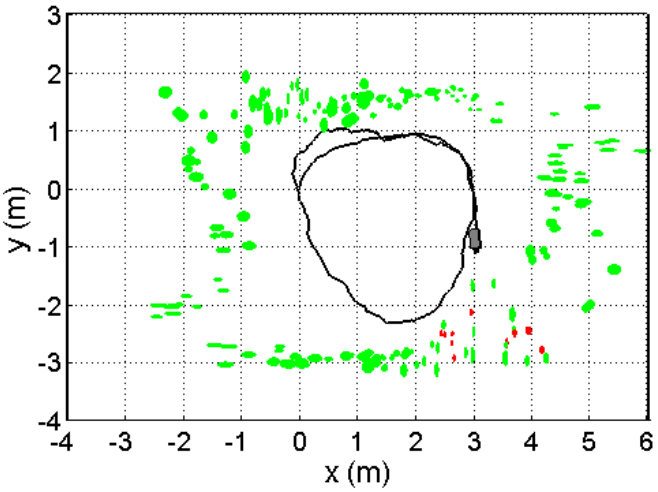


圖 3.26 850th 第二圈的行進中

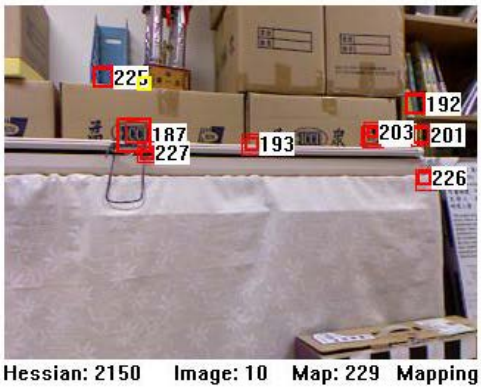


圖 3.27 1104th 彩色影像

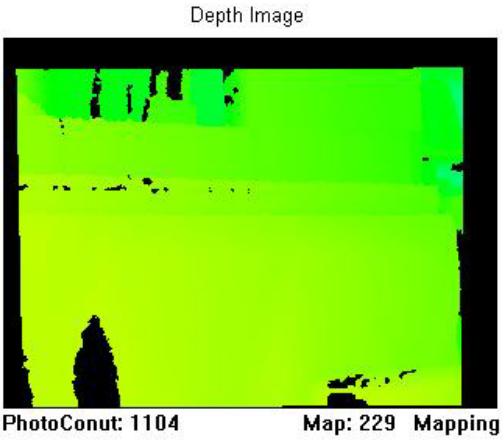


圖 3.28 1104th 深度影像

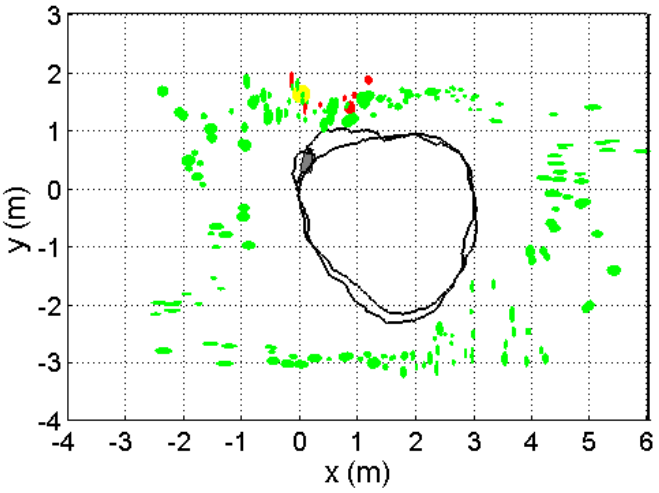


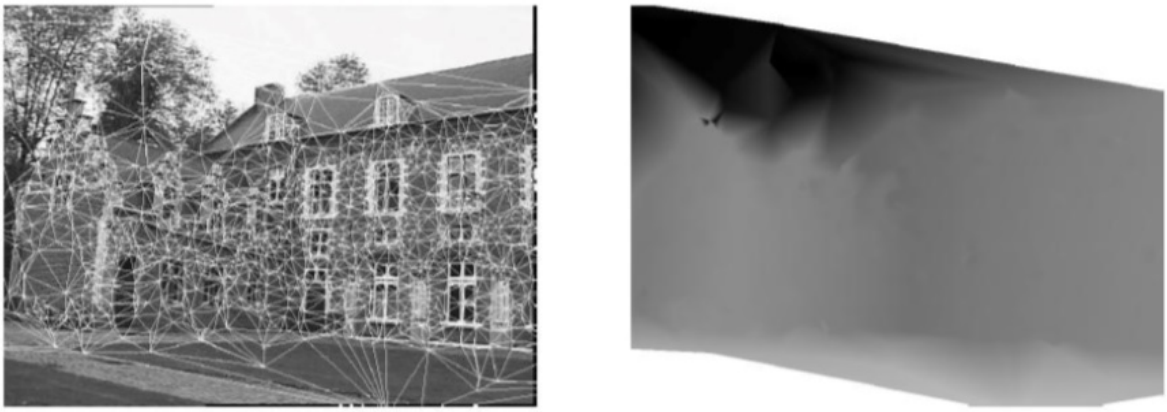
圖 3.29 1104th 路徑第二圈重合

第4章 Kinect SFM

4.1 場景重建(Structure from Motion)

SfM 方法[8]主要利用了影像中物體的移動量、相機運動以及物體在三維空間中運動等物理關係。SfM 最大的好處在於可以極佳的模擬針孔相機模型(pin-hole camera model)與 epipolar geometry 等物理現象。然而 SfM 不足的地方在於針對某些拍攝狀況(如僅有攝影機旋轉的運動)無法進行深度之估測。此外，SfM 也無法應用於非靜止畫面且場景中的物體會形變的狀況。但在一般的狀況下 SfM 法可以提供一個精確且不錯的深度圖。SfM 主要的目的為經由一組影像計算拍攝時的攝影機運動參數以及 3D 場景幾何資訊，影像可經由校正或非校正的攝影機拍攝。SfM 為電腦視覺的熱門研究領域，在文獻[9]中提出了一個基於 SfM 的深度估測演算法，其流程主要可以分為三個部分：(1)特徵追蹤(feature tracking)、(2)場景幾合重建(motion and structure recovery)與(3)深度估測(dense depth map creation usinggeometry fitting)。由於 SfM 是利用連續的幾張畫面估測場景的運動以及攝影機的運動等影像與識別幾何關係，這些關係可以透過計算各畫面中的特徵(特徵點、線、表面資訊等)之間的相關性，而特徵擷取與特徵追縱的演算法已有許多的文獻可以參考，計算出畫面間特徵的相關性後，便可應用於計算攝影機之移動以及場景的結構，最後再經由三角化(Delaunay triangulation)演算法計算出深度圖，如圖 4.1 所示。SfM 的演算法如前所述，受限於某些拍攝時的場景，因此無法應用於所有的情況，由其是在攝影機不動的狀況之下，可能因場景的資訊不足而無法計算深度，且特徵追蹤的運算複雜度高，對於即時轉換是一大挑戰，因此短期內較難實際商品化。

在本論文實驗中將以SURF[12]作為特徵追蹤，EKF SLAM[6][13][14]做為場景重建的依據，用Kinect深度資訊作為深度估測，來實現SfM的場景重建。



(a)三角化的結果　　(b)場景深度圖

圖 4.1 文獻 [9] 所使用的 SfM 深度估測結果

4.2 Kinect 感測器實現場景重建

SFM 演算法流程圖，如圖 4.2 所示；此演算法是由 Kinect SLAM 得到 BinoSLAM 檔、影像以及深度，用Matlab 讀取 Kinect SLAM 得到的資料後，迴圈開始讀取每張圖且計算出各像素的世界三維坐標，如方程式(4.1)，其中 m_i 特徵點相對世界坐標(world frame)的三維坐標，$\mathbf{m}_i = [X_i, Y_i, Z_i]^T$；$R_C^W$ 維攝影機坐標系{C}(carmera frame)相對於世界坐標{W}的旋轉矩陣(rotation matrix)，以相對於固定座標的基本旋轉(elementary rotations) [Sciavicco and Siciliano 1996]表示為方程式(4.2)，其中 $c\phi = \cos\phi$、$s\phi = \sin\phi$。ϕ_x、ϕ_y 與 ϕ_z 為攝影機相對於世界座標軸 x、y 與 z 的轉動角度。$h_i^C = [h_{ix}^C, h_{iy}^C, h_{iz}^C]^T$ 稱為視線向量；h_i^C 如方程式(4.3)，其中 (I_{ix}, I_{iy}) 為特徵點的影像平面座標值，單位為像素(*pixel*)；f_u 與 f_v 為攝影機 x 方向與 y 方向的焦距，單位為像素；(u_0, v_0) 為影像平面中心點座標；*depth* 為影像平面的深度值，單位為公尺(m)，將方程式(4.2)、方程式(4.3)代入方程式(4.1)

並做移項動作，求算出特徵點相對於世界坐標系的三維坐標如方程式(4.4)，此時，用類別形態存取每個特徵點的三維坐標、RGB 值以及 state，為了避免重複的點被重複畫過，所以再從這些資料裡，判斷與前一張的影像的各坐標做 $\Delta x, \Delta y, \Delta z$，如果 $\Delta x, \Delta y, \Delta z$ 的絕對值小於所設定的誤差值範圍時，state 的值就為 1，之後再判斷如果 state 為 0 時就畫圖，為 1 時就結束不畫圖，並回到迴圈的起點開始做下張圖片的 n 次迴圈，做完後得到場景重建如圖 4.3。

$$m_i = r + R_C^W \times h_i^C \tag{4.1}$$

$$R_C^W = \begin{bmatrix} c\phi_y c\phi_z & s\phi_x s\phi_y c\phi_z - c\phi_x s\phi_z & c\phi_x s\phi_y c\phi_z + s\phi_x s\phi_z \\ c\phi_y s\phi_z & s\phi_x s\phi_y s\phi_z + c\phi_x c\phi_z & c\phi_x s\phi_y s\phi_z - s\phi_x c\phi_z \\ -s\phi_y & s\phi_x c\phi_y & c\phi_x c\phi_y \end{bmatrix} \tag{4.2}$$

$$\begin{aligned} h_x &= \frac{(Ix - u0) \times h_z}{fu} \\ h_y &= \frac{(Iy - v0) \times h_z}{fv} \\ h_z &= depth \end{aligned} \tag{4.3}$$

$$\begin{aligned} m_x &= r_x + (cy \times cz) \times h_x + (sx \times sy \times cz - cx \times sz) \times h_y + (cx \times sy \times cz + sx \times sz) \times h_z \\ m_y &= r_y + (cy \times sz) \times h_x + (sx \times sy \times sz + cx \times cz) \times h_y + (cx \times sy \times sz - sx \times cz) \times h_z \\ m_z &= r_z + (-sy) \times h_x + (sx \times cy) \times h_y + (cx \times cy) \times h_z \end{aligned} \tag{4.4}$$

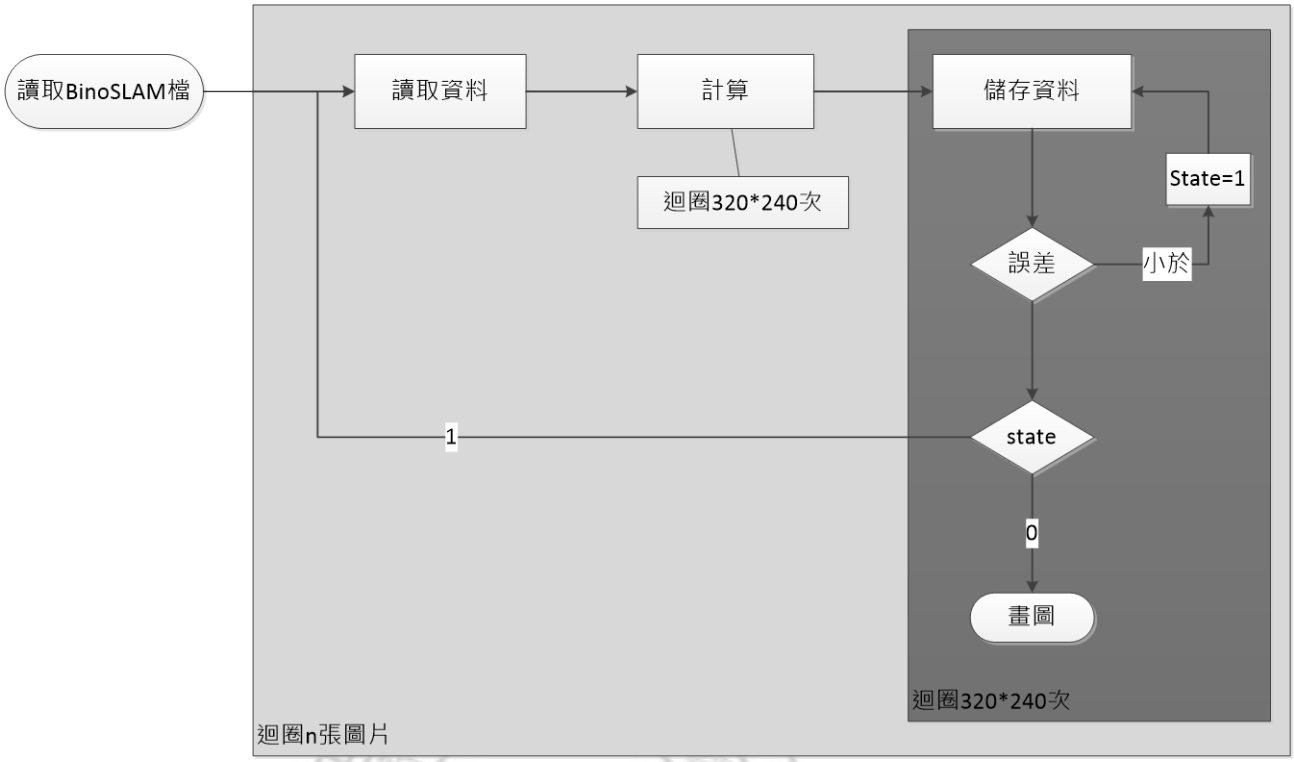


圖 4.2 場景重建流程圖

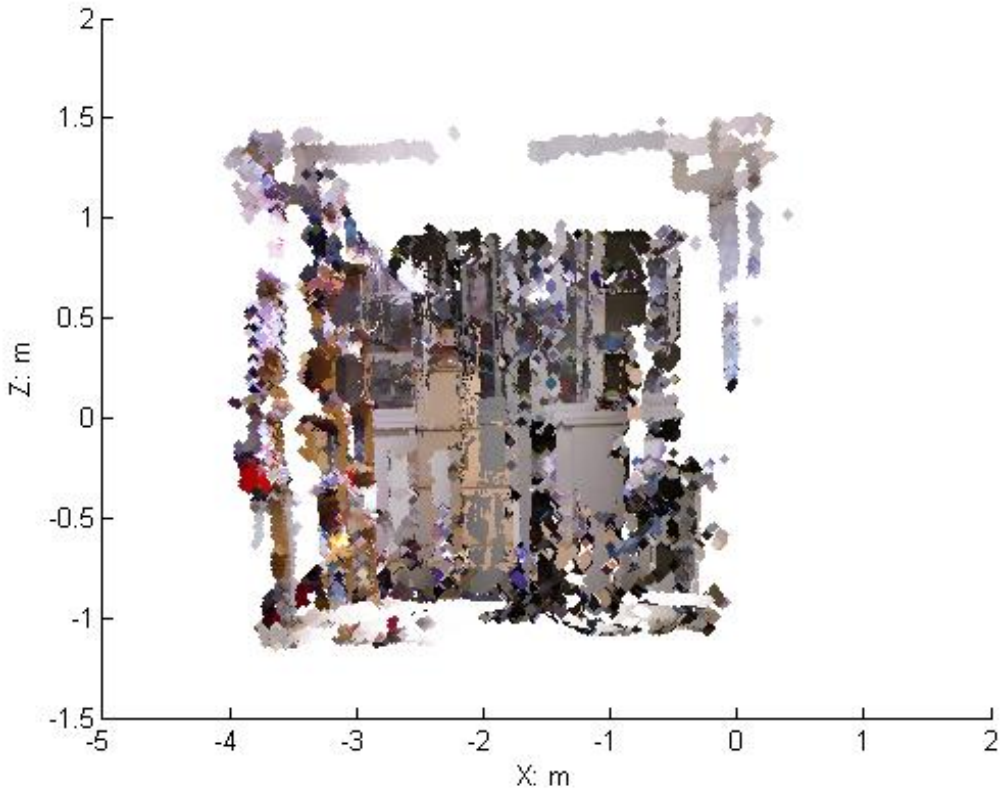


圖 4.3 場景重建圖場景重建

4.3 範例一：Kinect 感測器實現場景重建

本論文在上章節中利用了 EKF SLAM 得到了攝影機的三維座標，接下來本章實驗利用已得到的三維座標數據來做場景重建。將 EKF SLAM 得到彩色和深度影像，如圖 4.4 圖 4.5，以及用 Matlab 算出的路徑圖，如圖 4.6。再利用這些數據用 Matlab 做場景重建，如圖 4.7。以下為場景重建的過程之影像擷取：

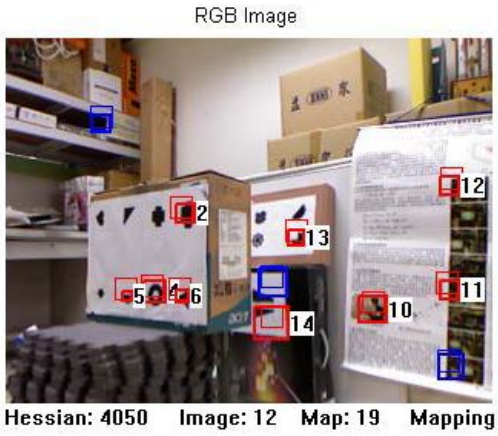


圖 4.4 100th 彩色影像

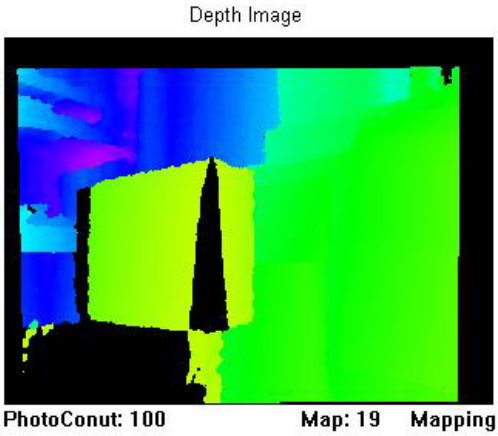


圖 4.5 100th 深度影像

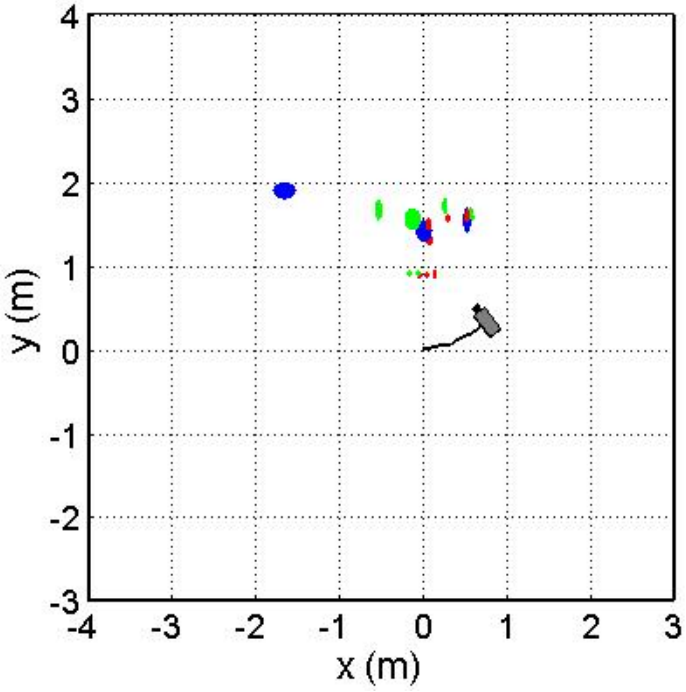


圖 4.6 100th 路徑規劃

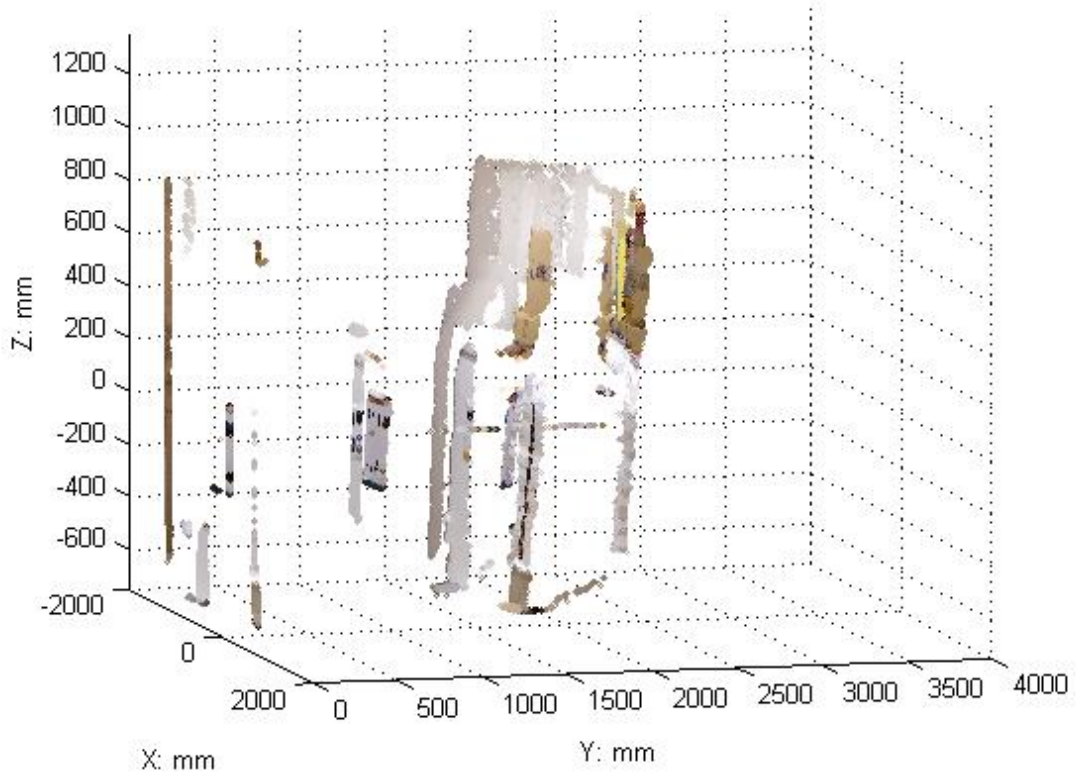


圖 4.7 100th 場景重建

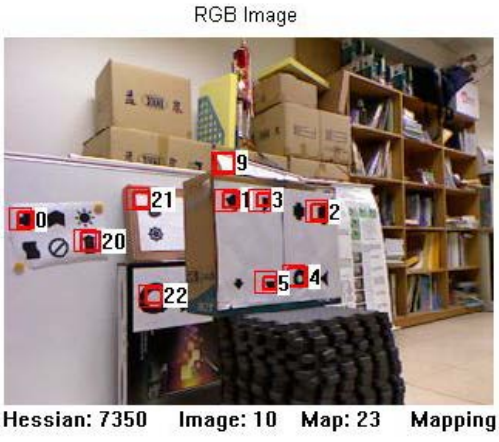


圖 4.8 300th 彩色影像

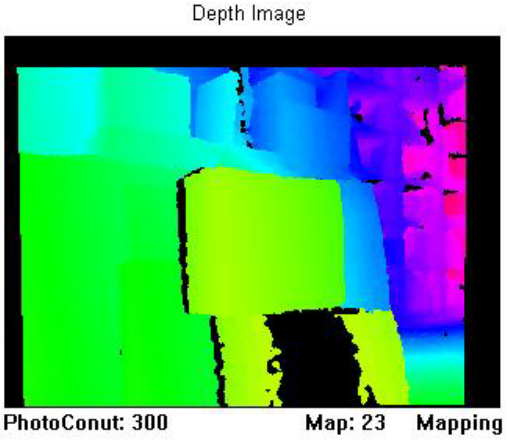


圖 4.9 300th 深度影像

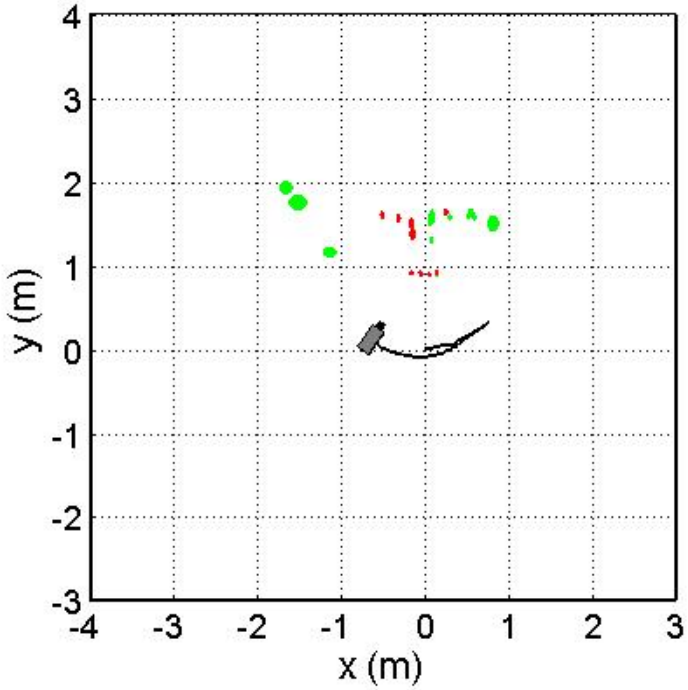


圖 4.10 300th SLAM 的路徑規劃

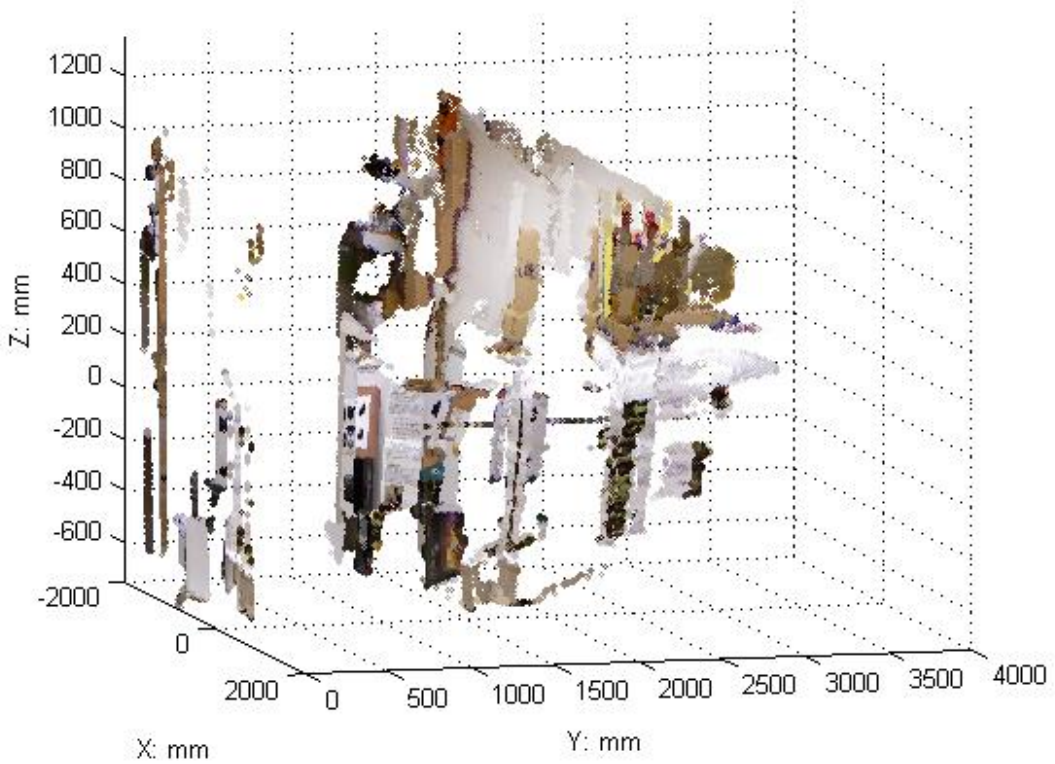


圖 4.11 300th SFM 場景重建

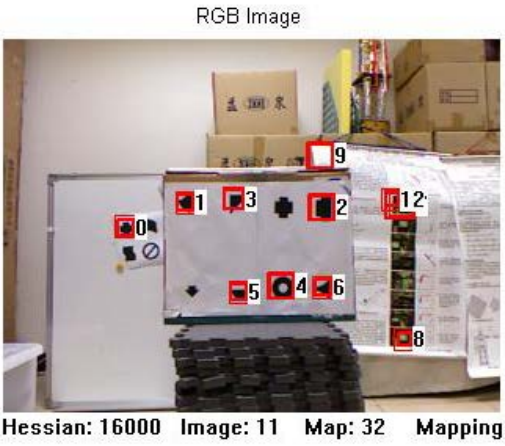


圖 4.12 766th 彩色影像

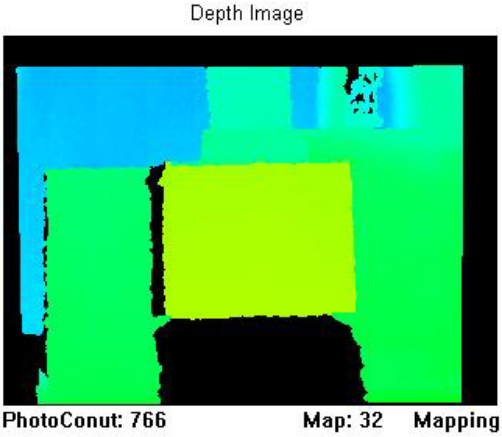


圖 4.13 766th 深度影像

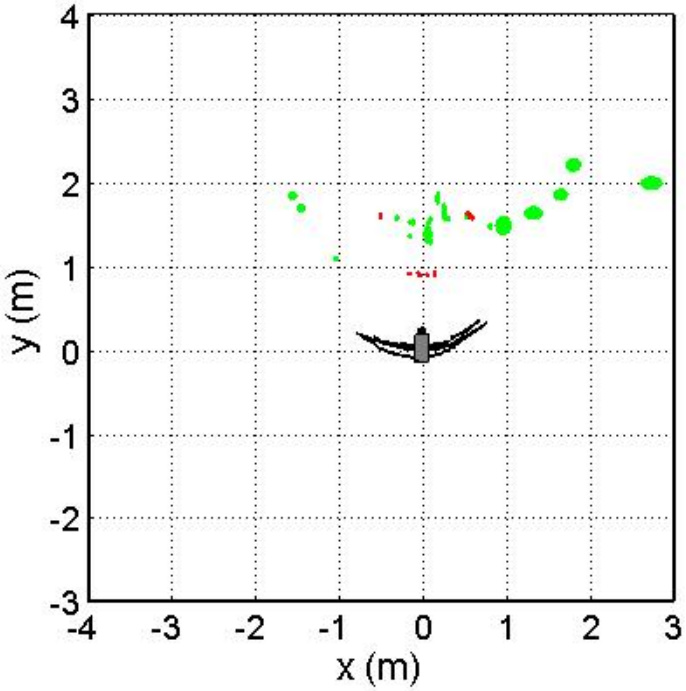


圖 4.14 766th SLAM 的路徑規劃

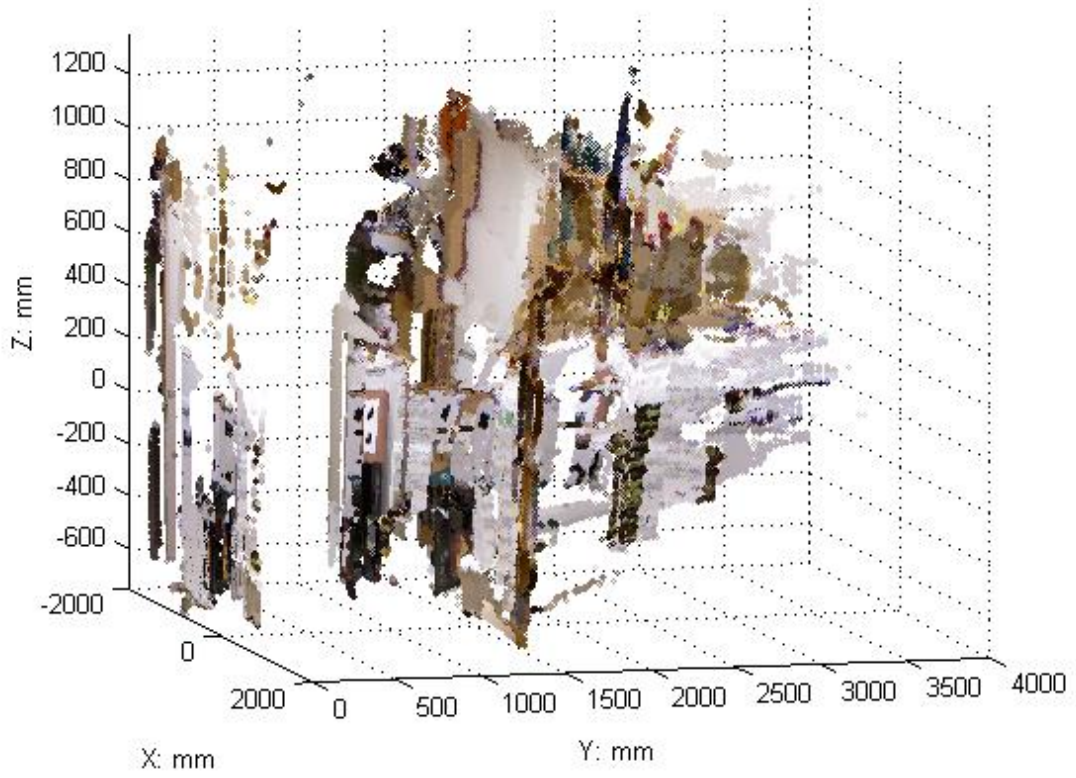


圖 4.15 766th SFM 場景重建

4.4 範例二：Kinect 感測器實現場景重建

此範例的目的是發現上一範例範例在場景重建之後會發生同樣的座標會有好幾個像素重複增加，根據此問題並增加了一個判斷機制，將排除相同的地方重複增加新的像素點，並增加場景重建的辨識度；相同的，在同一個場景去作場景重建，這次在實驗圖裡增加上正面和側面的角度；再以同樣的方法，用不一樣的場景作場景重建。以下為場景重建過程：

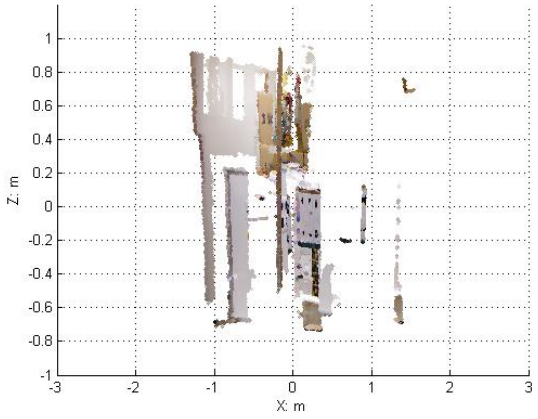


圖 4.16 100th 場景重建(正面)

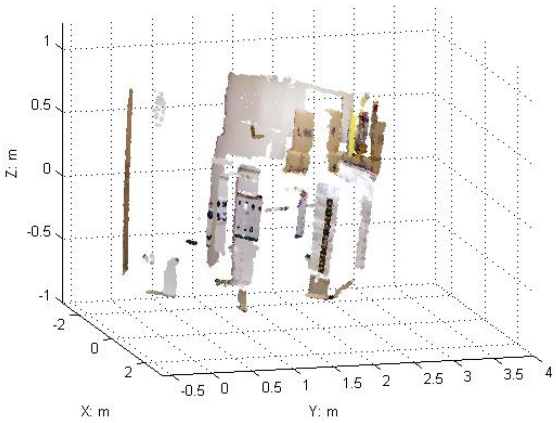


圖 4.17 100th 場景重建(側面)

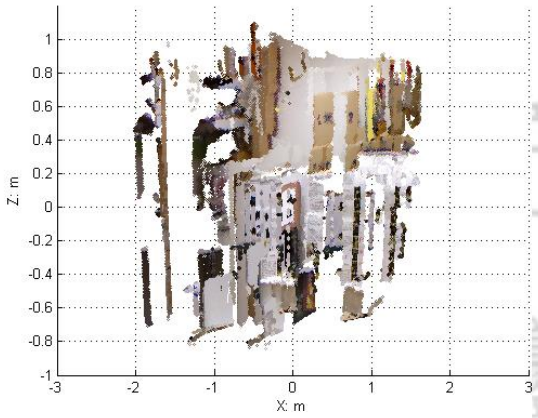


圖 4.18 300th 場景重建(正面)

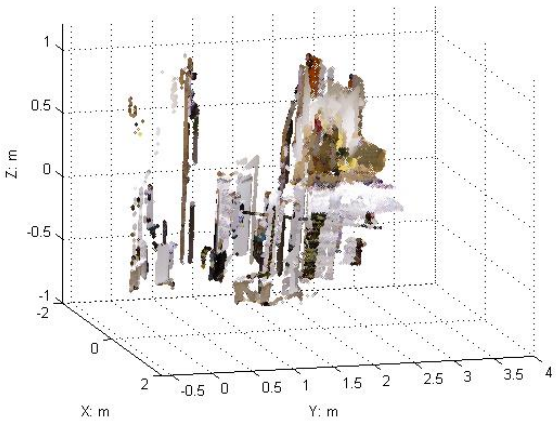


圖 4.19 300th 場景重建(側面)

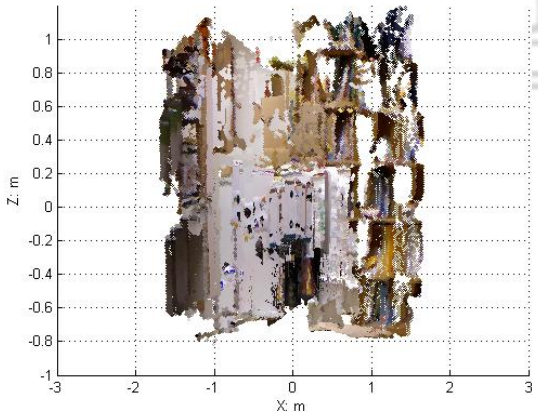


圖 4.20 766th 場景重建(正面)

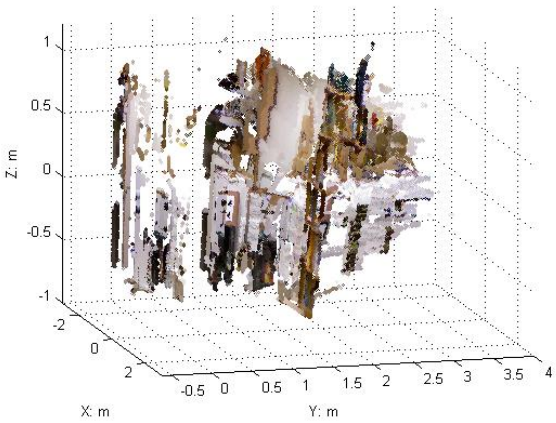


圖 4.21 766th 場景重建(側面)

4.5 範例三：Kinect 感測器實現場景重建

此範例將行走直線，目的是實驗以不同的實驗路徑劃出場景重建圖。

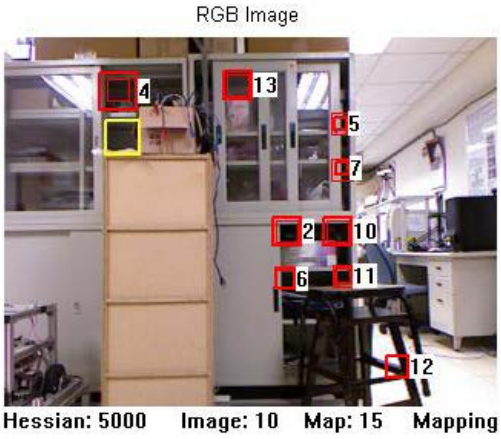


圖 4.22 100th 彩色影像

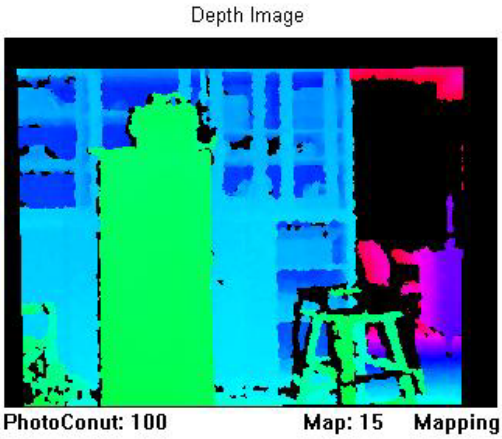


圖 4.23 100th 深度影像

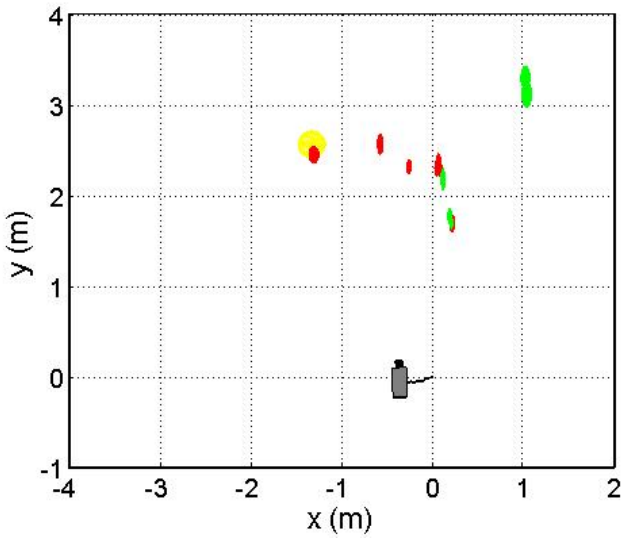


圖 4.24 100th SLAM 路徑規劃

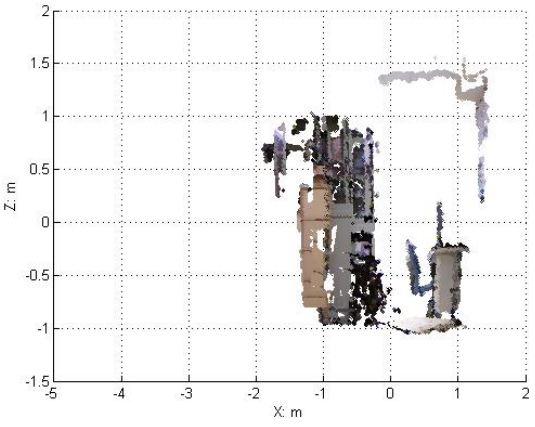


圖 4.25 100th 場景重建(正面)

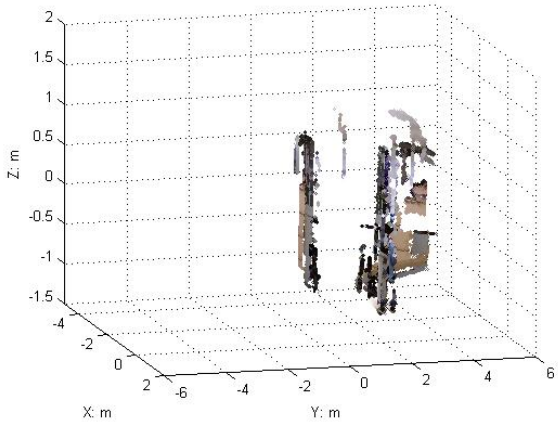


圖 4.26 100th 場景重建(側面)

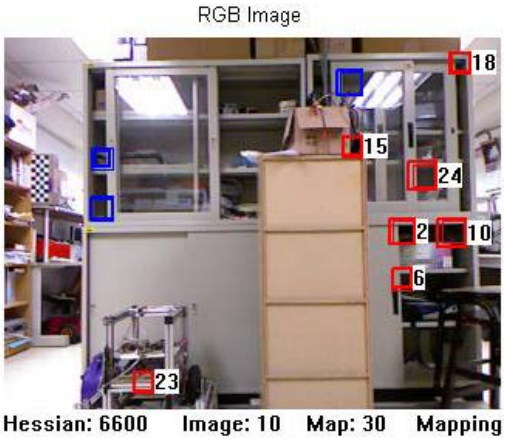


圖 4.27 200th 彩色影像

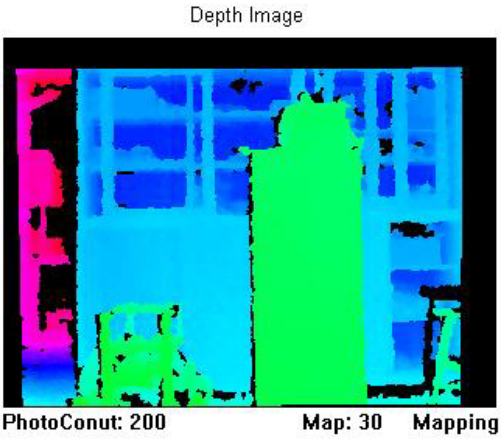


圖 4.28 200th 深度影像

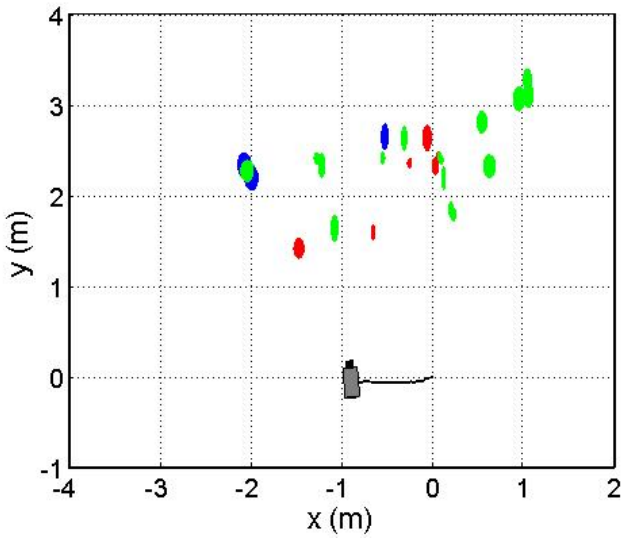


圖 4.29 200th SLAM 路徑規劃

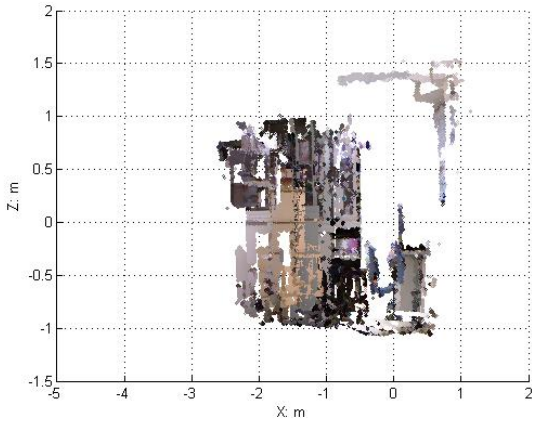


圖 4.30 200th 場景重建(正面)

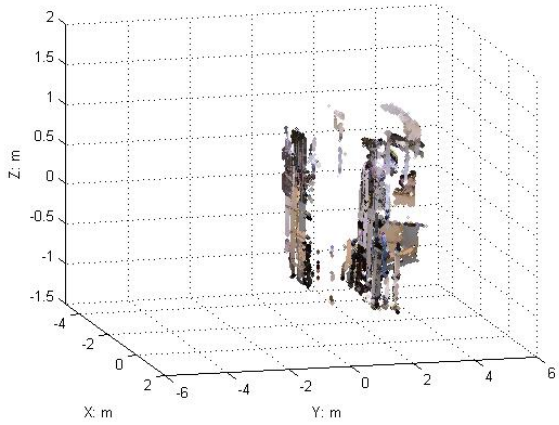


圖 4.31 200th 場景重建(側面)

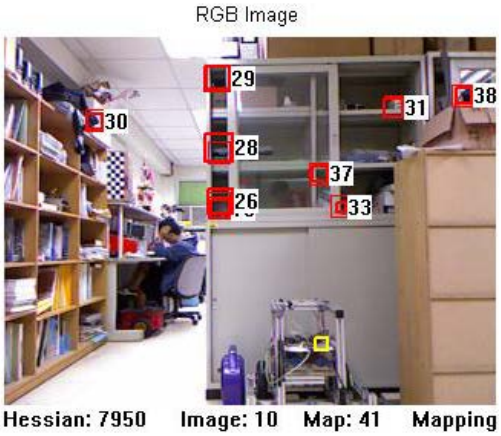


圖 4.32 300th 彩色影像

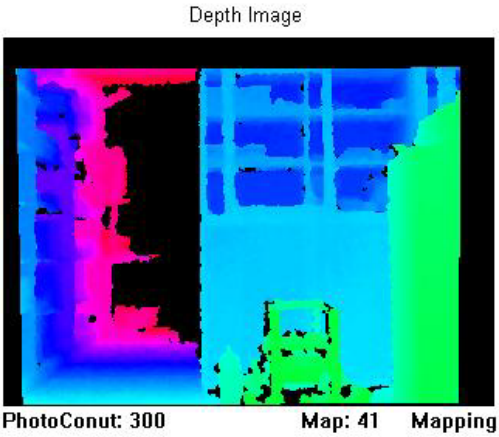


圖 4.33 300th 深度影像

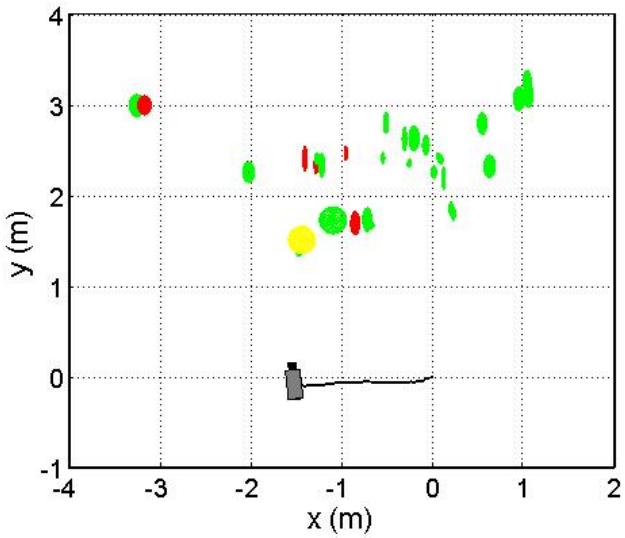


圖 4.34 300th SLAM 路徑規劃

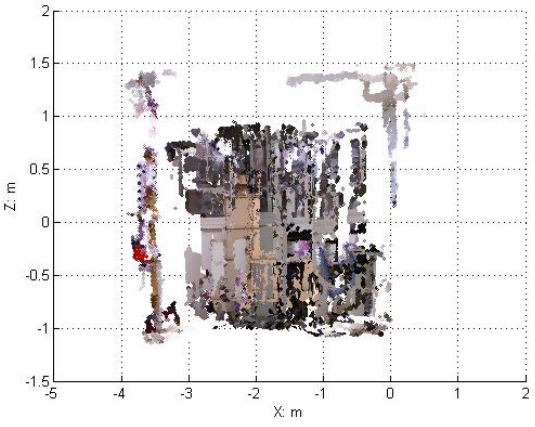


圖 4.35 300th 場景重建(正面)

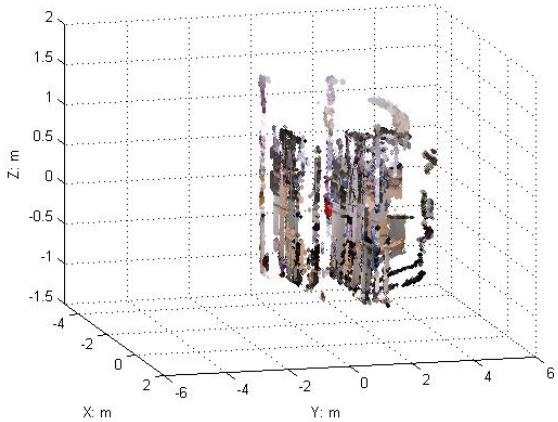


圖 4.36 300th 場景重建(側面)

第5章 雲端運算

5.1 雲端運算

雲端運算（Cloud Computing）[10]是利用於網際網路的方式，傳輸使用者分享的軟硬體資源和訊息，並按照需要提供給電腦和其他行動裝置。雲端運算在社會中也提供許多網際網路新的 IT 服務並增加、使用與交付之模式，經常是虛擬化的資源或是空間，而且擁有動態與易擴充之好處，譬如 Google 雲端平台、Dropbox 或是 Apple 公司的 Cloud 等。雲端運算是繼 1980 年代大型電腦到客戶端-伺服器的大轉變之後的又一種巨變。使用者不再需要了解「雲端」中基礎設施的細節，不必具有相應的專業知識，也無需直接進行控制。

由於 EKF SLAM 的演算法運算龐大，如果在同一台運算器上運算的話，會影響運算時所需時間，所以本研究使用雲端運算，如圖 5.1。希望能把運算時間縮短，提高實驗效率。雲端運算簡單來說，是用多台運算器經由網路連接數據，分別運算不同的演算法，分擔在同一台運算器運算多個演算法，來提高運算速度。

在本研究中，規劃以一台 PC 都做 Server 端，分別計算演算法再經由網路互相傳輸來暫時替代多台雲端運算，如圖 5.2。首先將 SURF 與 EKF SLAM 這兩大演算法分開成兩個程式，分別放在兩台 PC 去做啟動，把 SURF 運算好的特徵點的數據利用網路傳輸送到另一台 PC 去做 EKF SLAM 運算，此運算稱為雲端傳輸；本實驗使用微軟 Visual C++的 afxinet.h 標頭檔，資料使用 FTP[11]去傳輸，將其中一台 PC 先設定好 FTP 平台後，再利用以上提到的方法去實作，系統流程圖如圖 5.3。

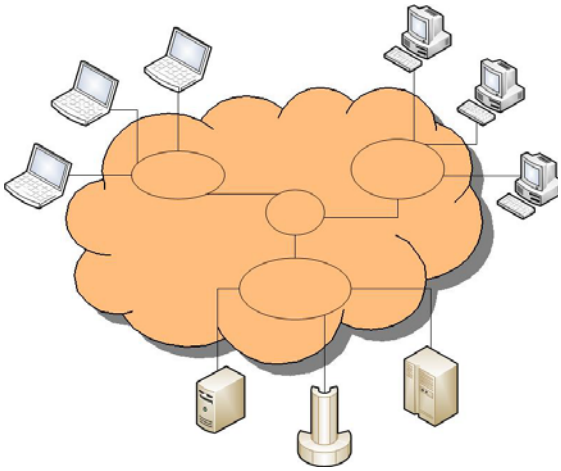


圖 5.1 雲端運算

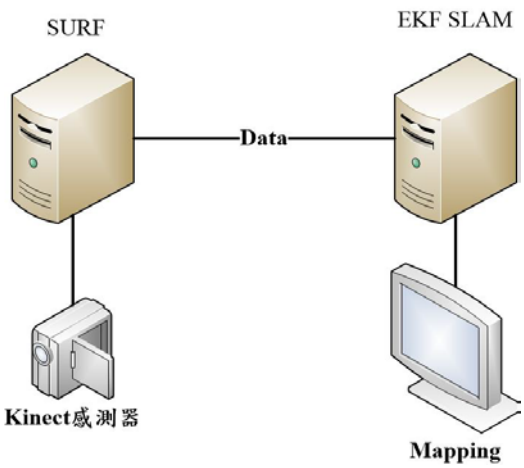


圖 5.2 兩 PC 簡單示意圖

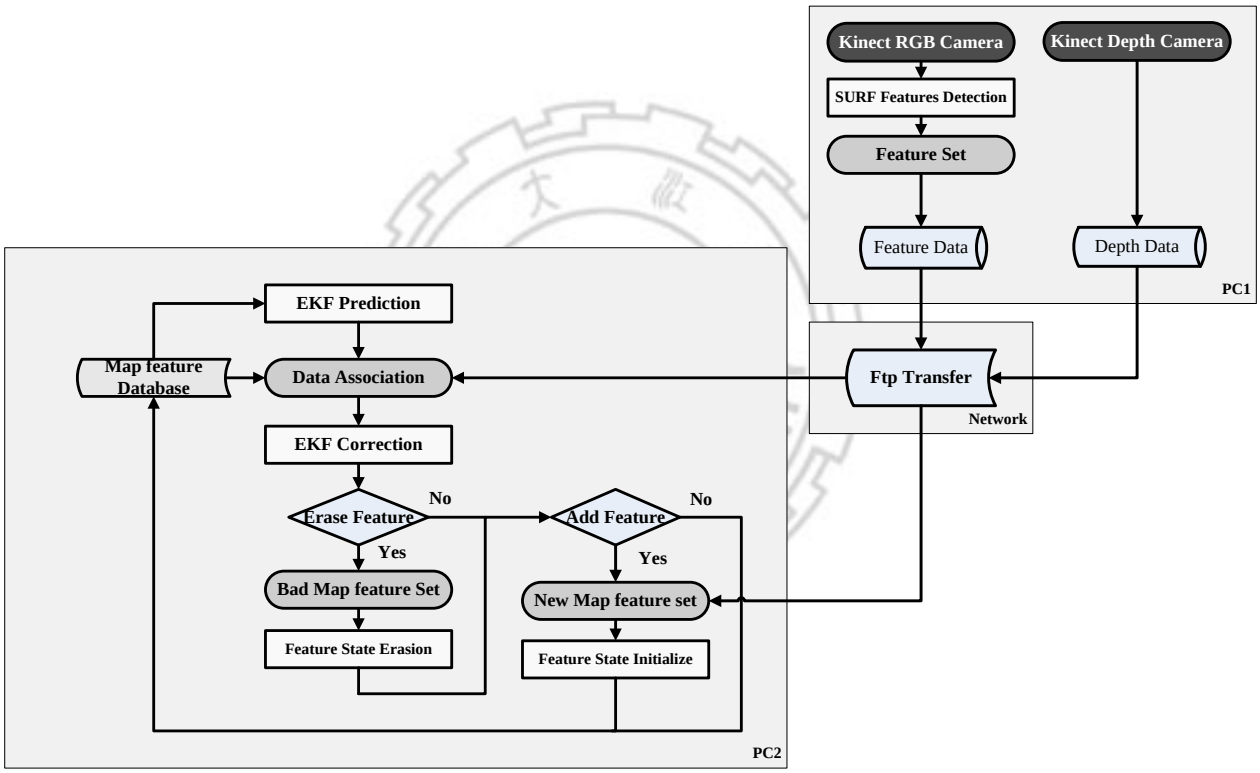


圖 5.3 雲端傳輸流程圖

5.2 範例一：模擬實驗場景為直線

本實驗將一台 PC 去做兩個運算式以及兩台 PC 去分別作兩個運算式的Δt 與相片張數用 Matlab 統計出結果，得其結果是兩台 PC 分別作兩個運算式的時間較短，並將時間比較用 Matlab 畫圖出來如圖 5.12，這樣的結果合乎本論文一開始的預期。之後再深入去找出為何會有時間的差異，並發現在做地圖規劃抓取特徵點的地方時，會產生延遲導致在一台 PC 為了要等上一步的運算式執行完而導致程式等待的時間，一樣的再用 Matlab 畫出各運算式的時間表，如圖 5.13；由於，在資料關聯(DataAssociation)這段時間花了相當多的時間，一樣再以此函式更細分出 SURF 與 Match 這兩個運算式，經由比較後 SURF 花的時間比 Match 花的時間久，如圖 5.14，這樣本實驗就可以解釋在只有一台 PC 花的運算時間會被 SURF 這個運算式的時間所影響，並導致一台 PC 程式較兩台 PC 程式運算較久的時間，由於兩台 PC 因為將運算式分開不會互相影響所以運算時間明顯較一台 PC 快;本實驗也分析這兩個方法的地圖大小與時間的比較，如圖 5.15、圖 5.16，很明顯的一台 PC 與兩台 PC 的地圖大小是差不多的，但時間相較之下是有差別的，所以這也說明雲端計算不影響實驗的結果，但卻又可以減少計算上的時間。

此節將以 EKF SLAM 的演算法作直線範例，並利用 Matlab 將一台 PC 與兩台 PC 的路徑、地圖大小以及時間做比較輸出成圖。首先，實驗抓取往前走完一次的彩色影像、深度影像、SLAM 路徑，如圖 5.4、圖 5.8；其次，以剛好走完一趟彩色影像、深度影像、SLAM 路徑，如圖 5.5、圖 5.9；再來，以走第二趟走完一次的彩色影像、深度影像、SLAM 路徑，如圖 5.6、圖 5.10；最後，以剛好走完第二趟的彩色影像、深度影像、SLAM 路徑，如圖 5.7、圖 5.11。

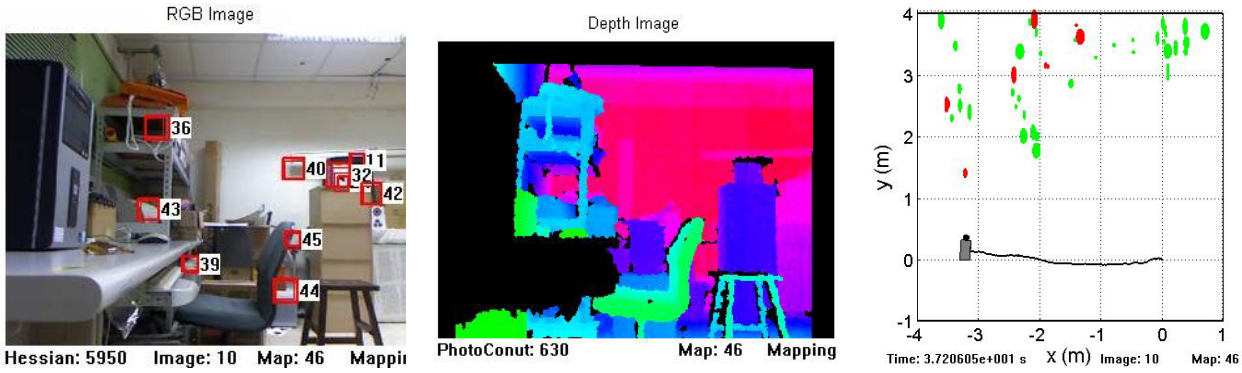


圖 5.4 1PC 630th 彩色影像、深度影像、與 SLAM 路徑

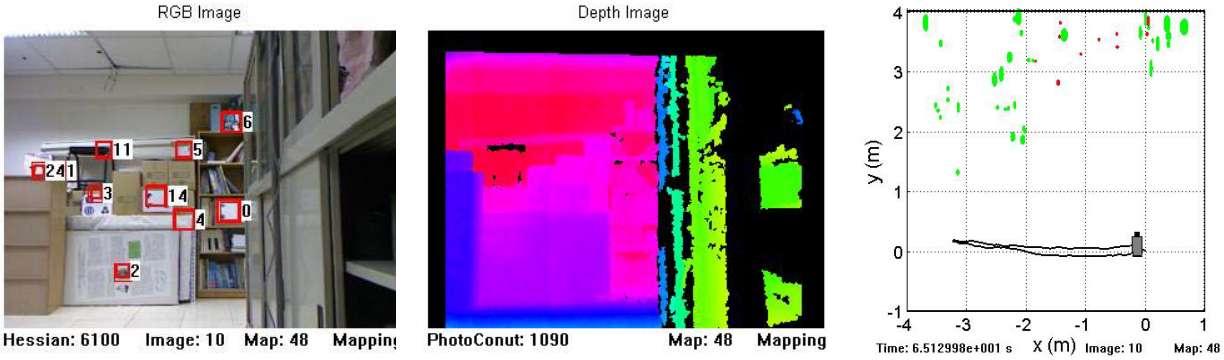


圖 5.5 1PC 1090th 彩色影像、深度影像、SLAM 路徑

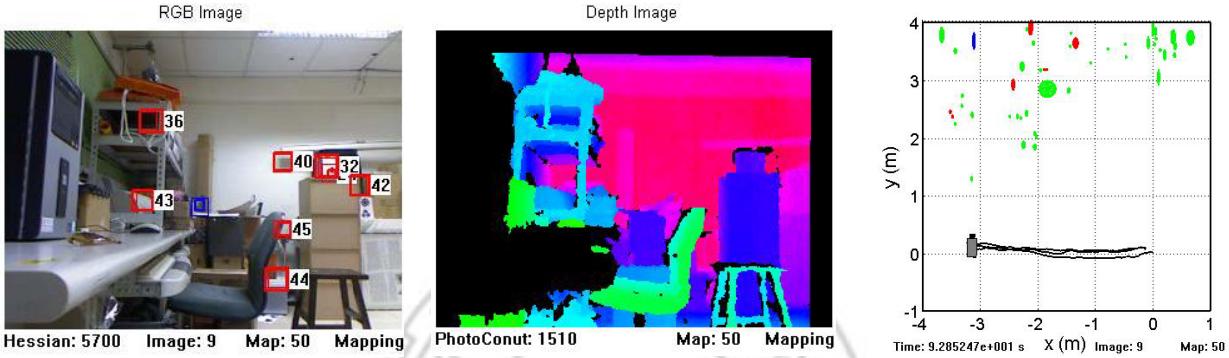


圖 5.6 1PC 1510th 彩色影像、深度影像、SLAM 路徑

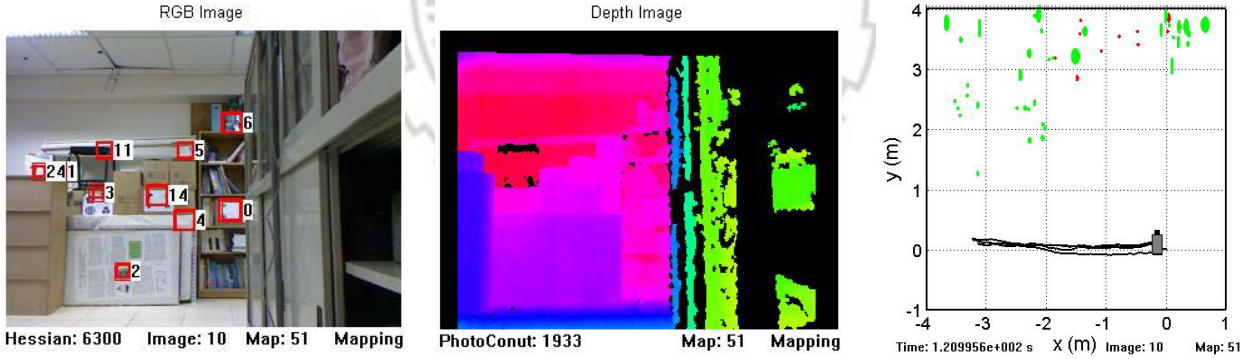


圖 5.7 1PC 1933th 彩色影像、深度影像、SLAM 路徑

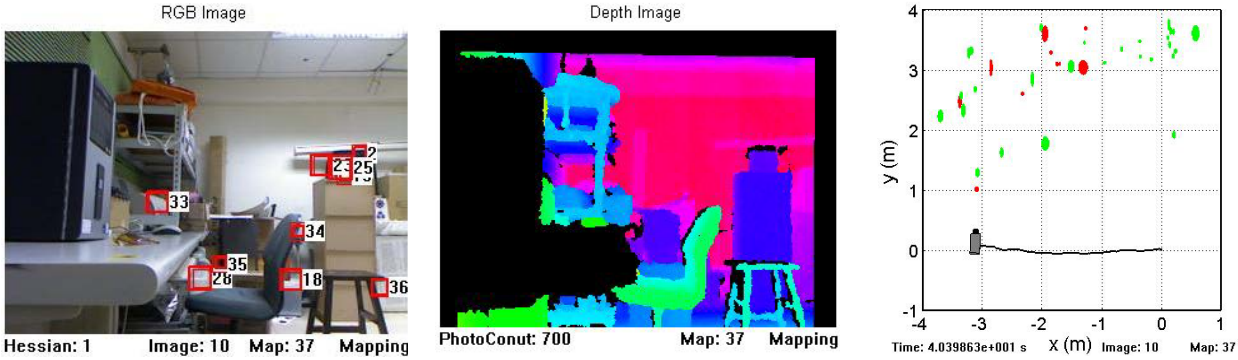


圖 5.8 2PC 700th 彩色影像、深度影像、SLAM 路徑

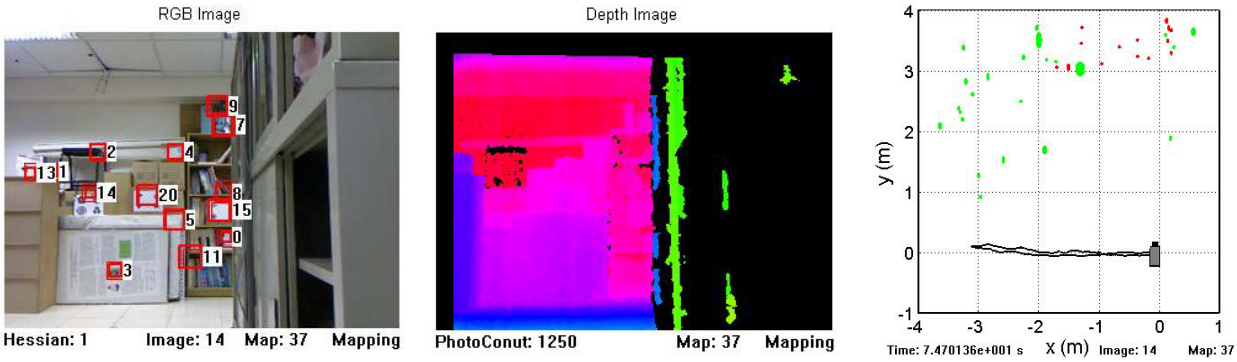


圖 5.9 2PC 1250th 彩色影像、深度影像、SLAM 路徑

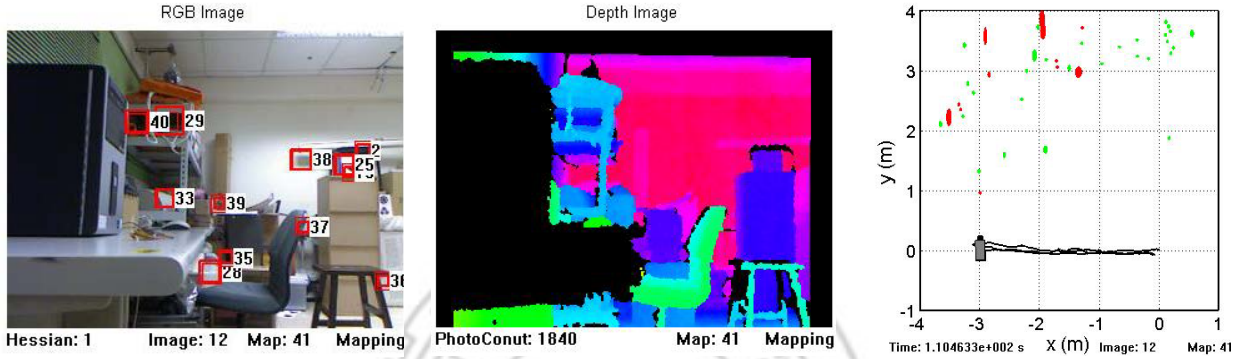


圖 5.10 2PC 1840th 彩色影像、深度影像、SLAM 路徑

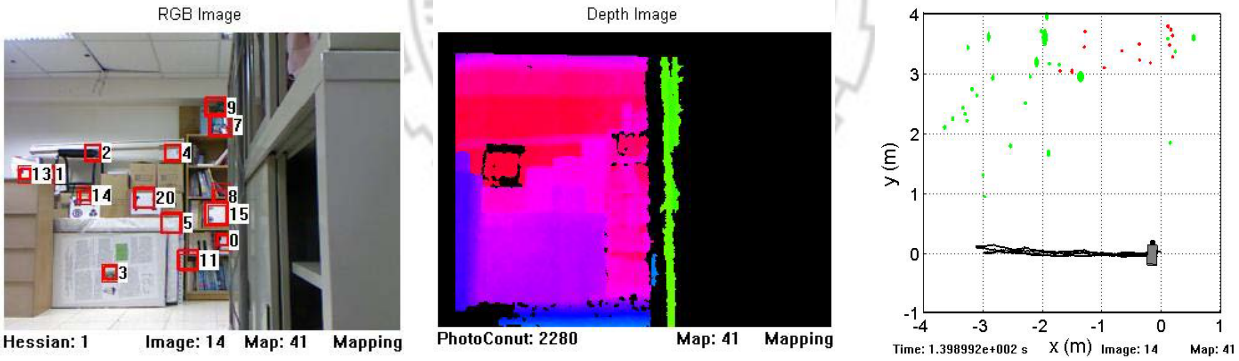


圖 5.11 2PC 2280th 彩色影像、深度影像、SLAM 路徑

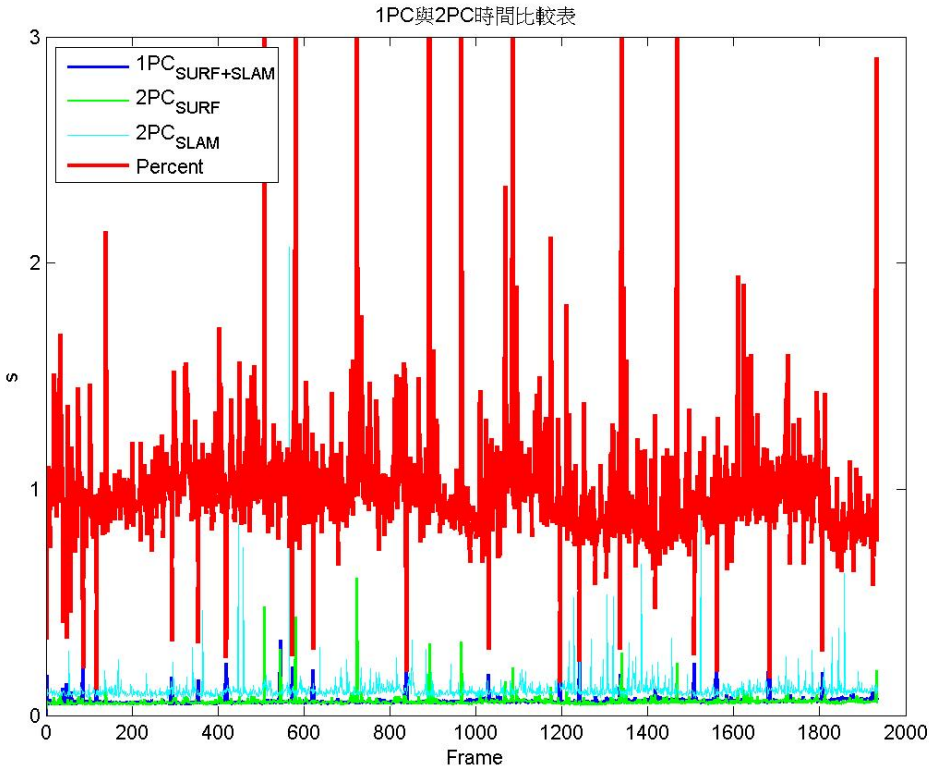


圖 5.12 範例為直線時，1PC 和 2PC 時間比較表

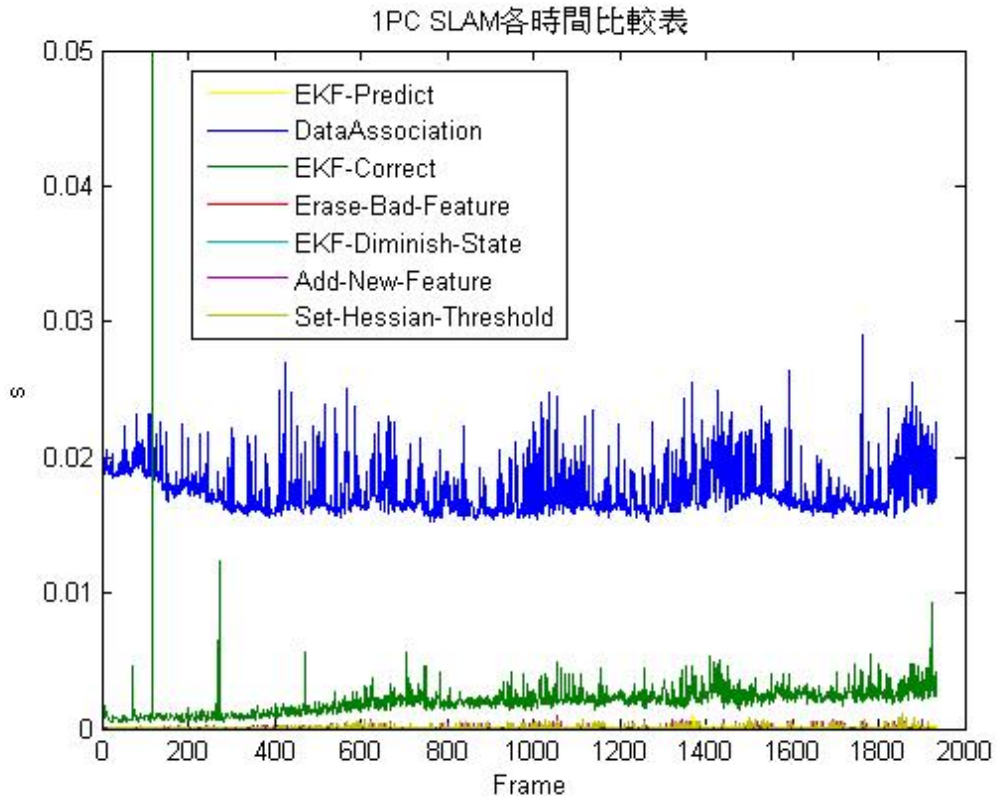


圖 5.13 範例為直線時，SLAM 各運算時間比較圖

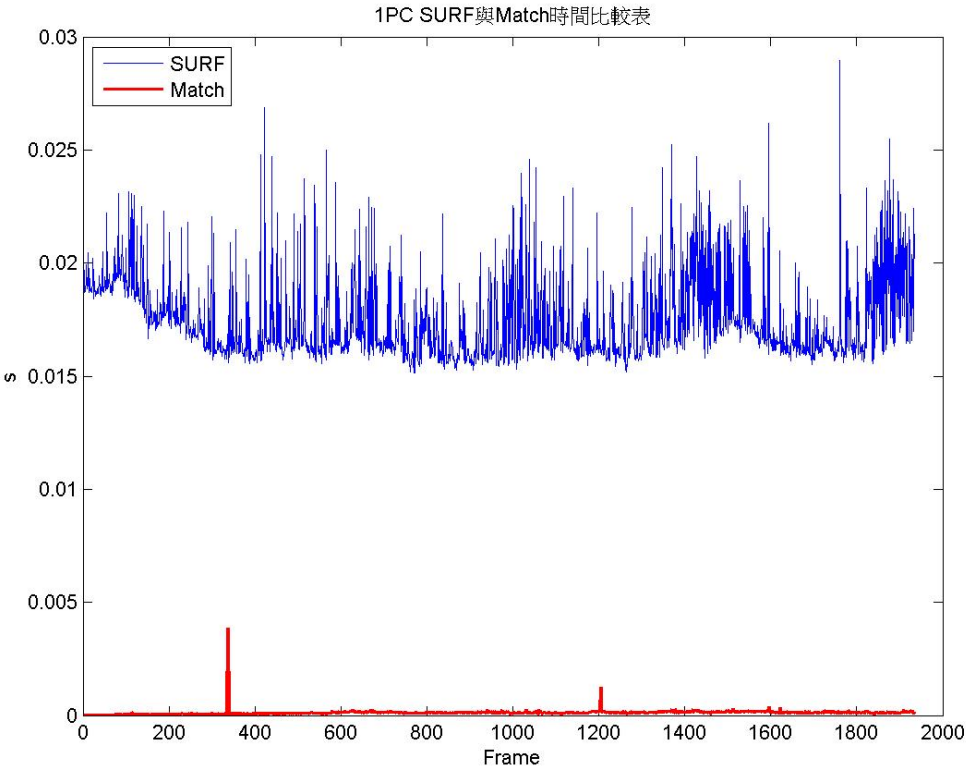


圖 5.14 範例為直線時，一台 PC SURF 與 Match 的時間比較

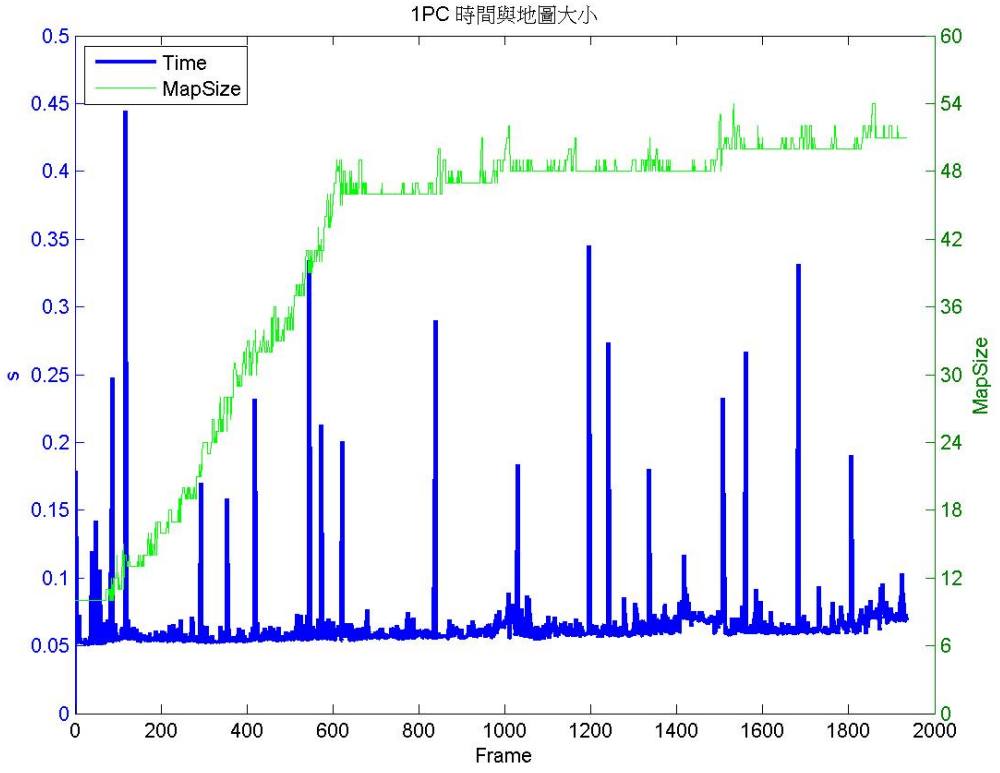


圖 5.15 範例為直線時，一台 PC 運算時間與地圖大小的比較

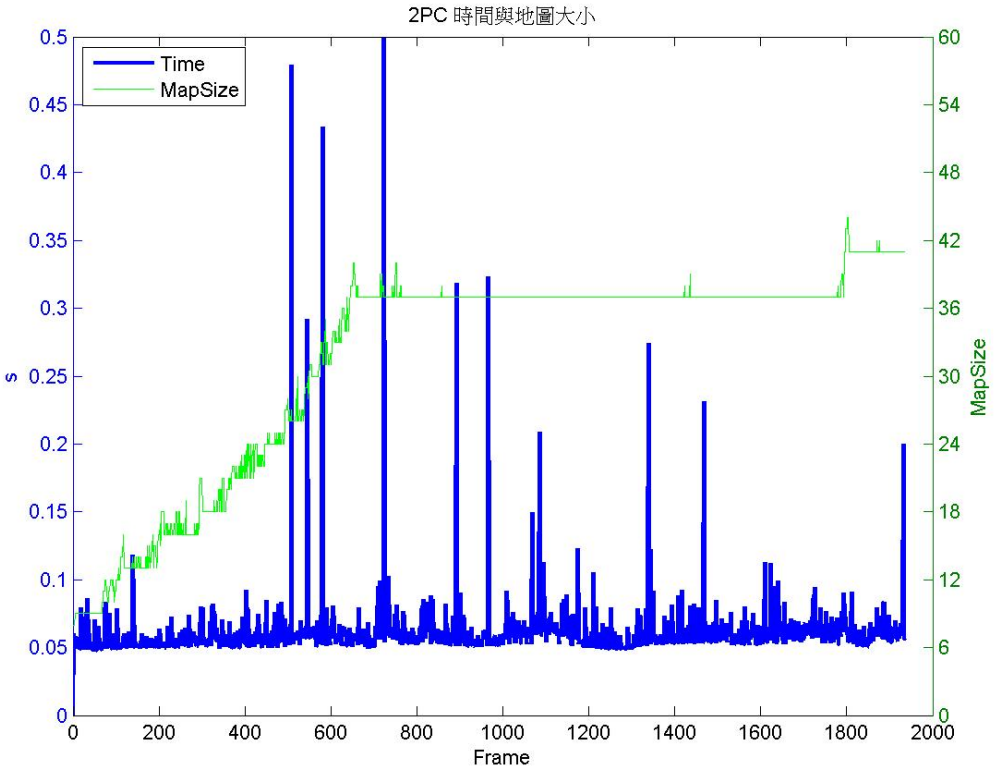


圖 5.16 範例為直線時，兩台 PC 運算時間與地圖大小的比較

5.3 範例二：模擬實驗場景為繞圈

利用和上一個範例一樣的方法，將兩個不同的方法用 Matlab 統計出結果，總時間比較表下，如圖 5.23，發現以長時間比較下來，在一台 PC 上的運算時間會越來越久，而在兩台 PC 上的運算時間幾乎沒有變化或是變少；再來，觀察一台 PC 的各運算式的時間，如圖 5.24，發現在跑長時間下來，因為特徵點會不斷新增，所以在更新地圖(EKF_Correct)的地方時間會越來越久，導致總體運算時間增長；與上個範例一樣，在資料關聯(DataAssociation)是花最久的時間，所以再將此函式細分為 SURF 與 Match 兩個運算式，如圖 5.25，一樣的 SURF 的運算時間比 Match 運算時間要久，一樣解釋了在一台 PC 總運算時間會被 SURF 運算式時間所影響，所以導致一台 PC 的時間運算的比兩台 PC 的時間久；最後，分析這兩個方法的地圖大小與時間的比較，如圖 5.26、圖 5.27，一樣很明顯地在一台 PC 與兩台 PC 的地圖大小是差不多的，但時間上一台 PC 運算時間會一直在增加，而兩台 PC 幾乎沒改變，與上個範例結論一樣，使用 EKF SLAM 的演算法時，在兩台 PC 上執行會比一台 PC

執行的較快。

此節將以 EKF SLAM 的演算法作繞圈範例，並利用 Matlab 將一台 PC 與兩台 PC 的路徑、地圖大小以及時間做比較輸出成圖。首先，實驗抓取往前走第一次轉彎的彩色影像、深度影像、SLAM 路徑，如圖 5.17、圖 5.20；其次，以剛好走完一圈的彩色影像、深度影像、SLAM 路徑，如圖 5.18、圖 5.21；最後，以跑完兩圈的彩色影像、深度影像、SLAM 路徑，如圖 5.19、圖 5.22。

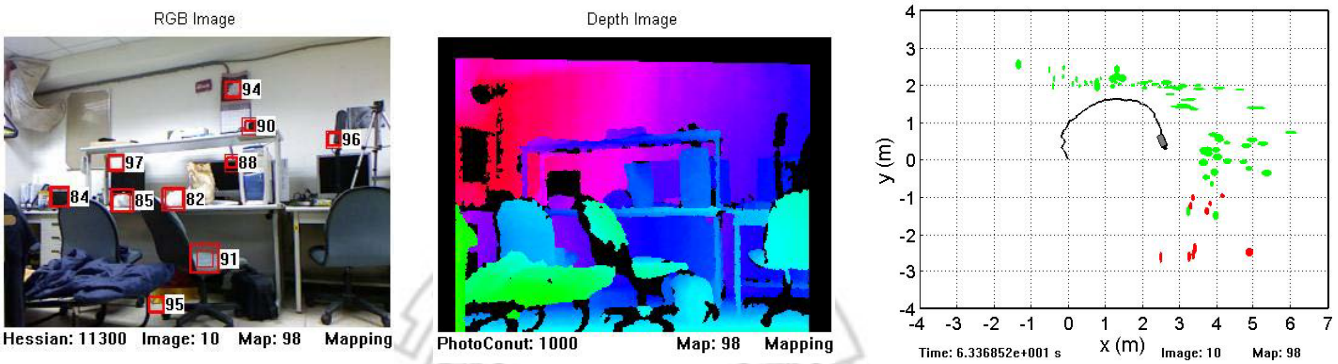


圖 5.17 1PC 1000th 彩色影像、深度影像、SLAM 路徑

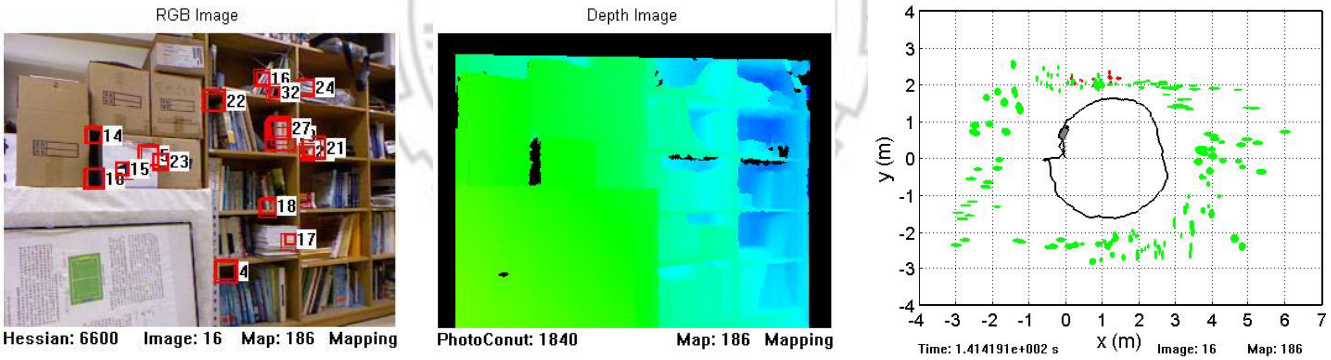


圖 5.18 1PC 1840th 彩色影像、深度影像、SLAM 路徑

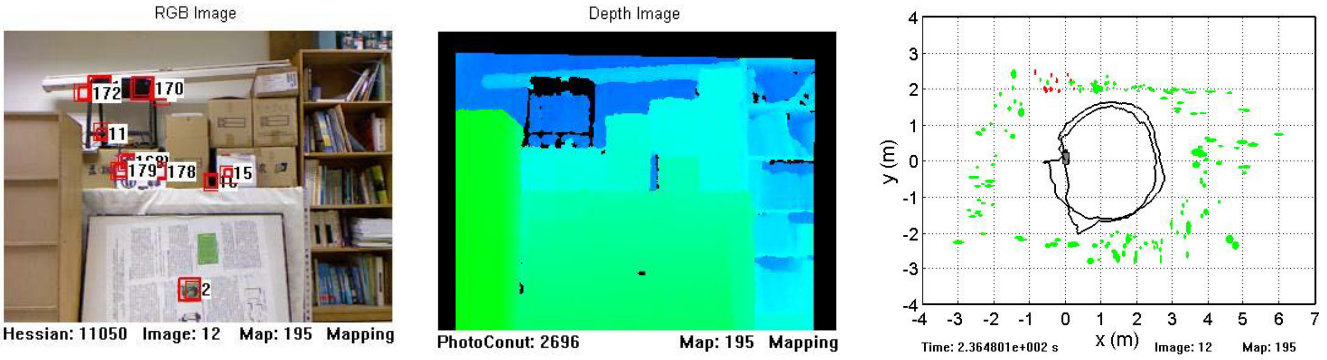


圖 5.19 1PC 2696th 彩色影像、深度影像、SLAM 路徑

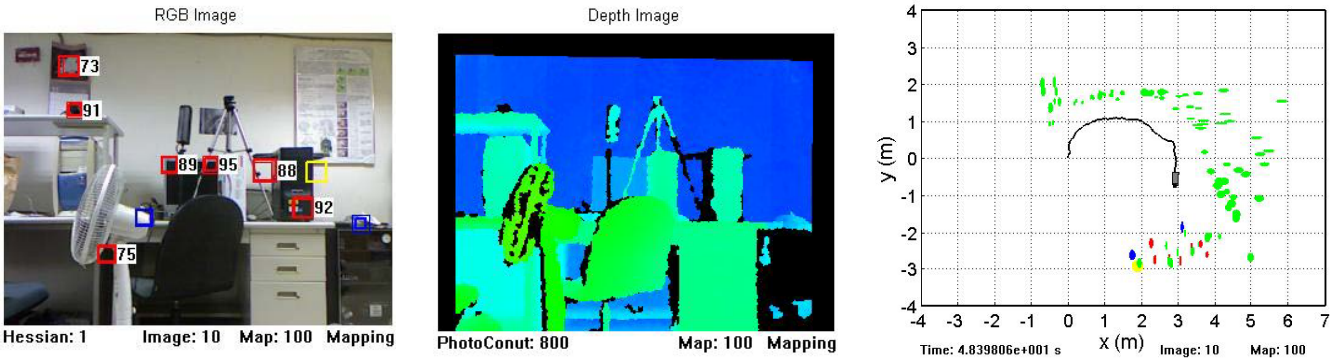


圖 5.20 2PC 800th 彩色影像、深度影像、SLAM 路徑

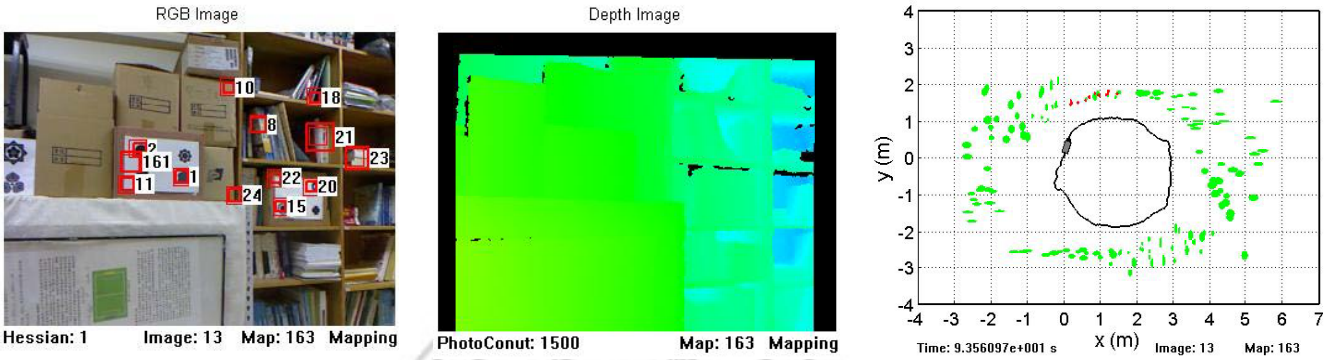


圖 5.21 2PC 1500th 彩色影像、深度影像、SLAM 路徑

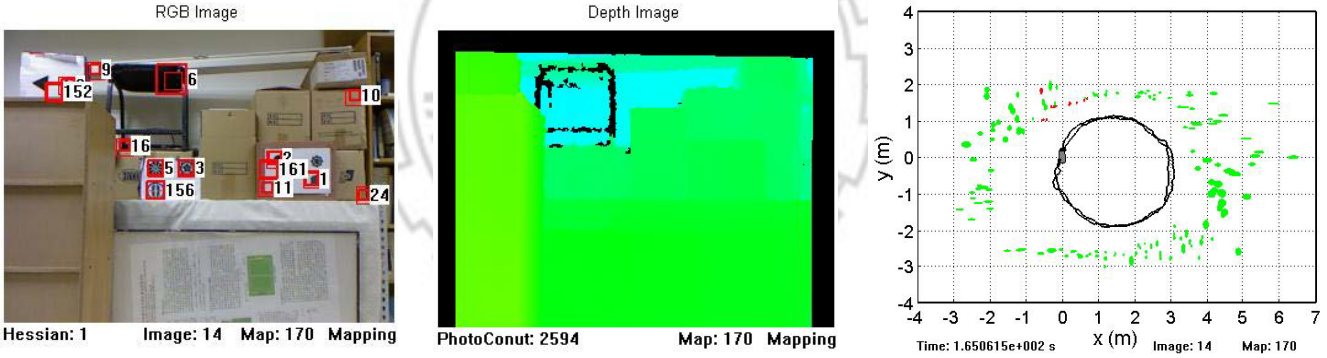


圖 5.22 2PC 2594th 彩色影像、深度影像、SLAM 路徑

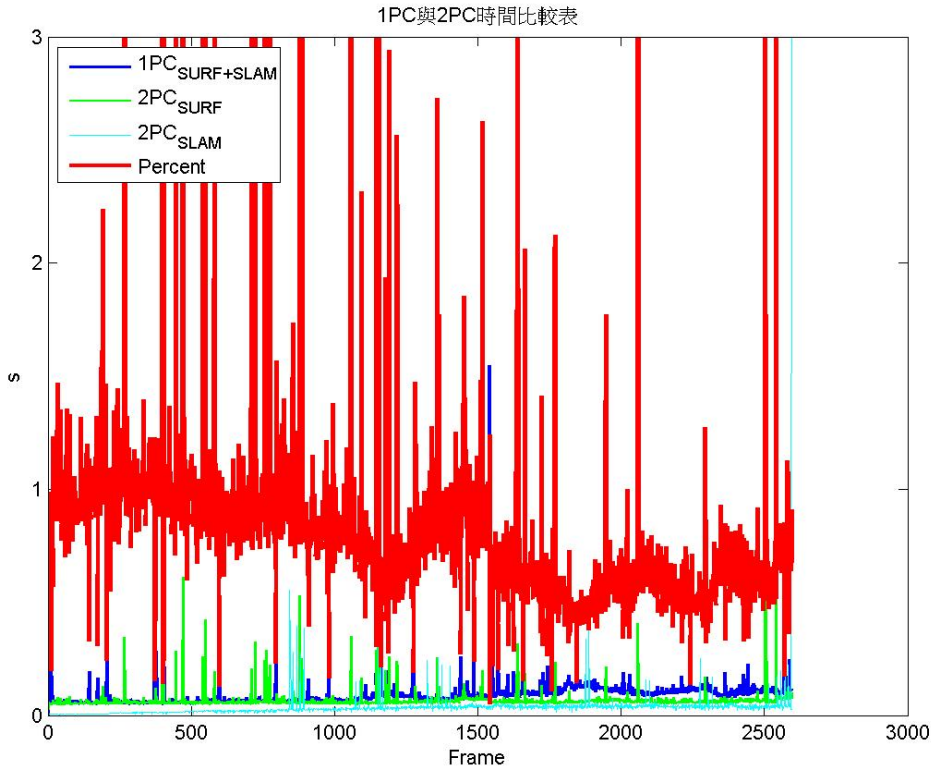


圖 5.23 範例為繞圈時，1PC 和 2PC 時間比較表

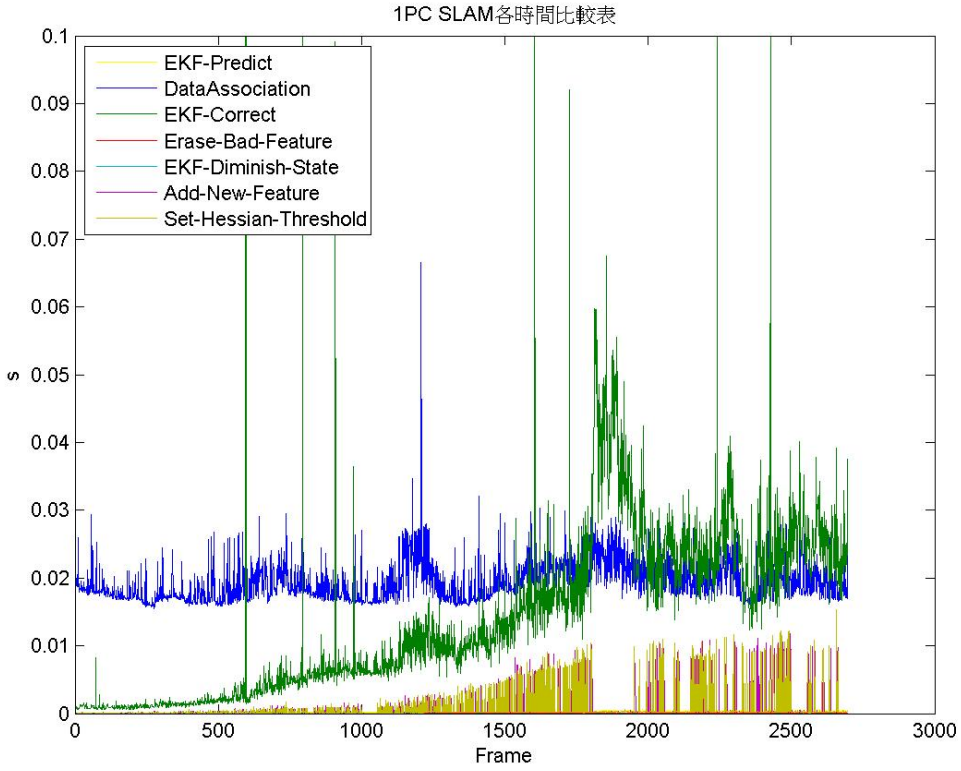


圖 5.24 範例為繞圈時，SLAM 各運算時間比較圖

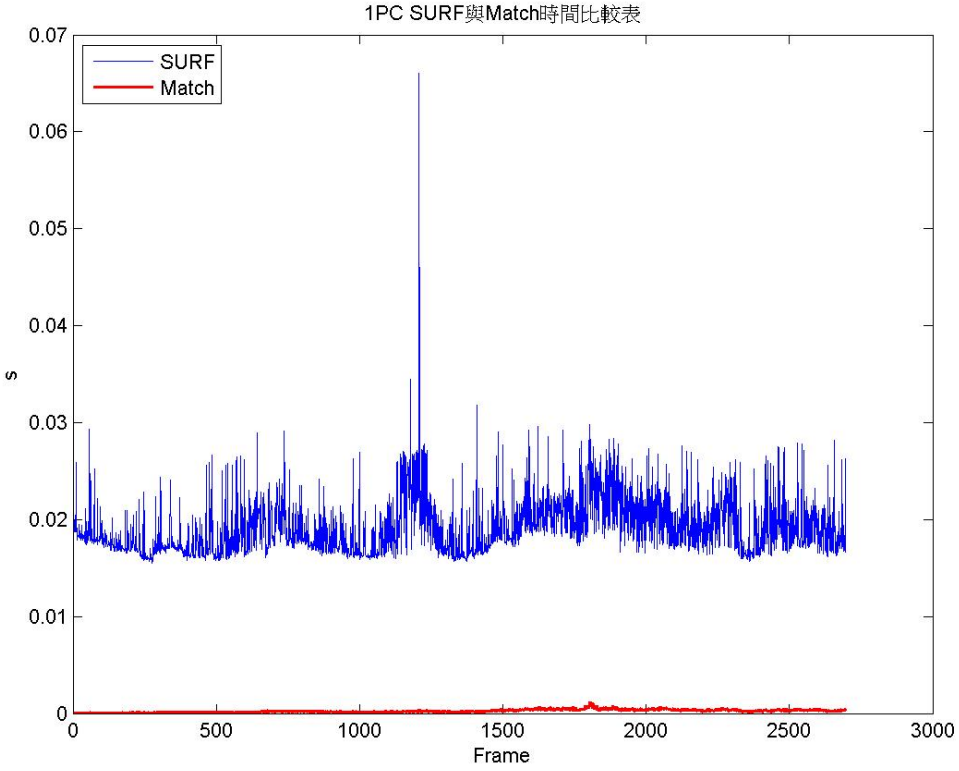


圖 5.25 範例為繞圈時，一台 PC SURF 與 Match 的時間比較

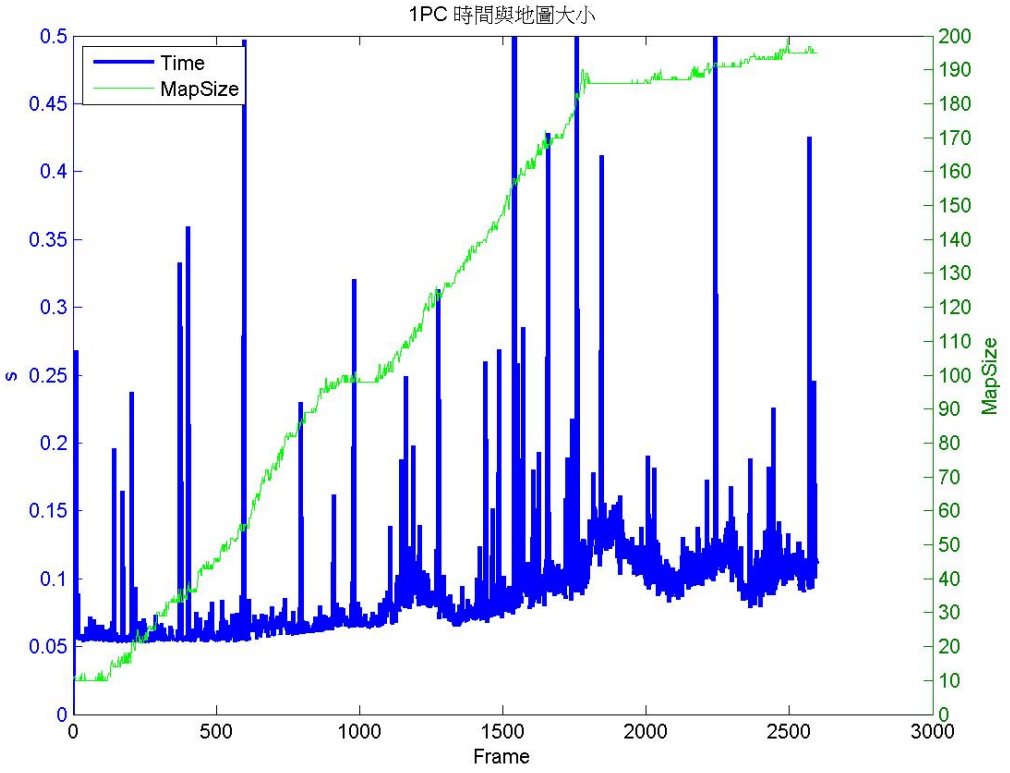


圖 5.26 範例為繞圈時，一台 PC 運算時間與地圖大小的比較

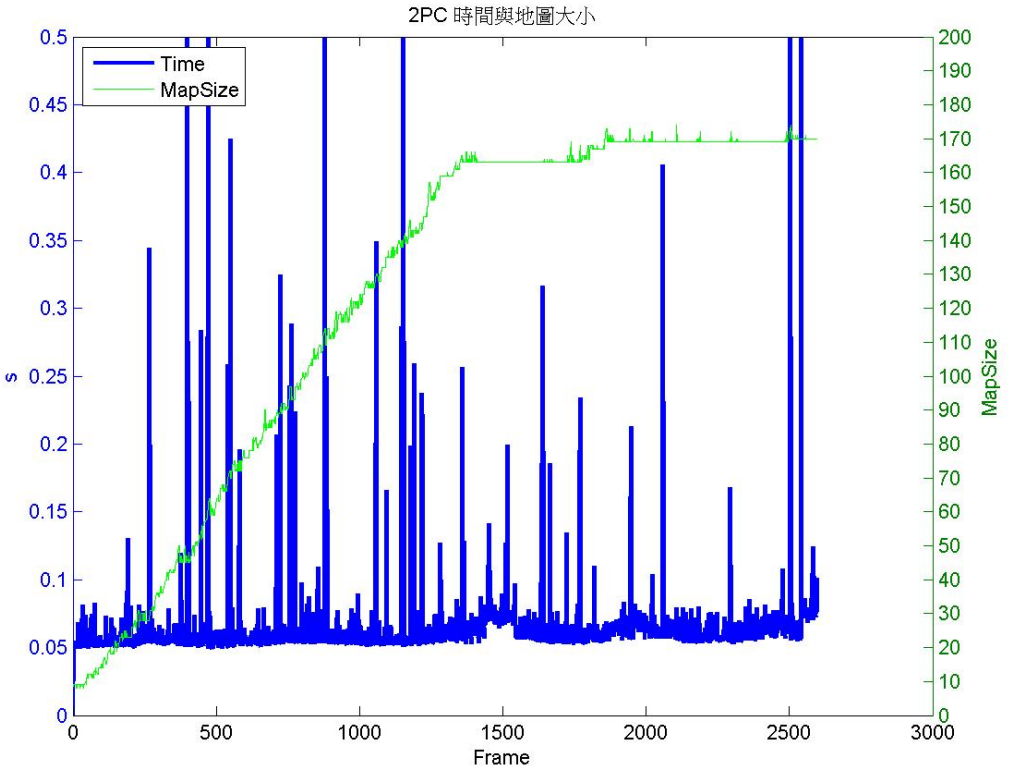


圖 5.27 範例為繞圈時，兩台 PC 運算時間與地圖大小的比較

第6章 研究成果與未來研究方向

6.1 研究成果

本論文規劃並實現 Kinect 做為 SLAM 感測器，並且測試 SLAM 和場景重建範例。完成建立 Kinect EKF SLAM、場景重建以及雲端運算等系統。

規劃 Kinect 感測器實現於 SLAM 系統上，完成的工作項目包括:

a. 使用歪斜校正去校正 Kinect 感測器的彩色影像與深度影像；
b. 實現用 Kinect 感測器去做同時定位與建圖的演算法；
c. 實測 Kinect 感測器，將 Kinect EKF SLAM 實現於室內環境的巡航任務上，完成直線和閉合的路徑測試。

規劃同時定位與建圖下的場景重建與雲端計算等系統，完成的工作項目包括:

a. 規劃場景重建的演算法；
b. 建立雲端計算的系統；
c. 完成場景重建的範例；
d. 實現未加雲端計算與加上雲端計算的 SLAM 系統，並進行量化比較。

本論文在機器人是同時定位與建圖的研究領域有三個具體的貢獻:

a. 使用 Kinect 感測器搭配 SURF 特徵，實現 EKF SLAM 之巡航任務；
b. 增加場景重建的功能，於巡航任務中建立場景模型；
c. 加入雲端計算的系統，減少同時定位與建圖的運算時間。

6.2 未來研究方向

a. 建立更完整 Kinect 感測器校準方法及解釋。
b. 將場景重建功能架構在機器人 SLAM 任務中實現，從 Kinect 所擷取豐富訊息中，選取場景重建所需資料。
c. 修改 Kinect 量測誤差所引起的特徵三維位置錯誤，提供場景重建的正確空間訊息，以建立正確的三維場景。
d. 建立雲端伺服器加入 SLAM 系統，並連結機器人控制器與雲端伺服器，發展連結所需的網路程式，將 SURF 特徵向量與像素等量測訊息傳送給雲端伺服器。
e. 在雲端伺服器上發展 SLAM 的 EKF 估測器與地圖管理等演算法的程式。

參考文獻

[1] Kinect for Windows,MSDN,website:http://msdn.microsoft.com/zh-tw/hh367958.aspx

[2] Kinect for Windows,Microsoft,website: http://www.microsoft.com/en-us/kinectforwindows/

[3] Bouguet, J.-Y., Camera Calibration Toolbox for Matlab, http://www.vision.caltech.edu/bouguetj/calib_doc/. (accessed 08/01/2010)

[4] Heikkila, J., and O. Silven, A Four-step Camera Calibration Procedure with Implicit Image Correction, *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp.1106-1112, 1997.

[5] Zhang, Z., A flexible new technique for camera calibration, *Microsoft Research Technical Report* 98-71, 1998. [http://research.microsoft.com/~zhang]

[6] Gonzalez R.C., and R.E. Woods, 2001, Digital Image Processing, Prentice Hall.

[7] 馮盈捷，使用尺度與方向不變特徵建立機器人視覺式 SLAM 之稀疏與續存性地圖，淡江大學機械與機電工程學系碩士論文，2011。

[8] 賴文能、陳韋志，國立中正大學電機工程研究所，淺談 2D 至 3D 視訊轉換技術。

[9] Ping Li, Rene Klein Gunnewiek, “On Creating Depth Maps from Monoscopic Video using Structure from Motion,” Proc. of IEEE Workshop on Content Generation and Coding for 3D-television, pp.508-515, 2006.

[10] Cloud-Computing,website: http://en.wikipedia.org/wiki/Cloud_computing

[11] FTP,website:http://en.wikipedia.org/wiki/File_Transfer_Protocol

[12] Bay, H., A. Ess, T. Tuytelaars, L. Van Gool, “SURF: speeded up robust features,” *Computer Vision and Image Understanding*, vol.110, pp.346-359, 2008.

[13] 洪敦彥，基於擴張型卡爾曼過濾器的機器人視覺式同時定位、建圖、與移動物體追蹤，淡江大學機械與機電工程學系碩士論文，2010。

[14] 邱明璋，基於極線限制條件之單眼視覺式移動物體偵測與追蹤，淡江大學機械與機電工程學系碩士論文，2011。