第三章

系統整合

第二章談到由於飛行模擬機座艙本身並無氣動力存在，所以力感裝置在飛行虛擬實境裡爲一個重要的裝置，該設備能讓使用者在飛行虛擬場景中擁有力回饋以逼近真實的駕駛感，所以將力感裝置順利地整合至飛行虛擬場景之中是一件非常重要的課題。本研究爲了實踐整合力感裝置之飛行模擬系統，提出系統架構示意圖如圖 3.1 所示[3,24]，當駕駛員施力推動力感裝置時，其力感裝置會產生操控命令送至飛行虛擬場景中。由於飛行虛擬場景端一方面具有計算飛行動態之區塊，可隨時配合駕駛員的操控以更新場景影像資訊，並經由顯像裝置將其影像輸出，此一迴路我們在此便稱之爲影像迴圈 (外迴圈)；另一方面虛擬場景端也具有搖桿力感模型的運算單元，可隨時比照駕駛員移動搖桿的角度而換算出參考之力命令送至力回饋控制器中做處理，並經由控制器的補償使力感裝置對駕駛員施與反饋力，則此一迴路我們在此便稱之爲力迴圈 (內迴圈)。

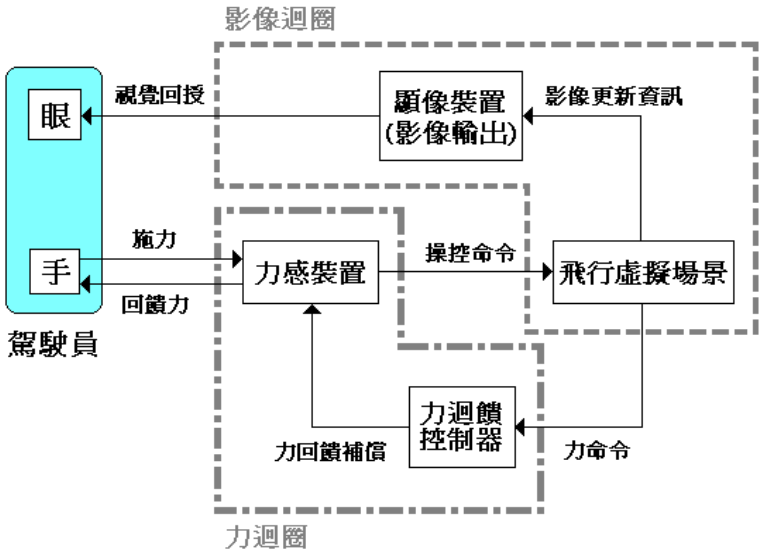

圖 3.1　整合力感裝置之飛行模擬系統示意圖

對於整合系統架構了解後，我們由圖 3.2 更進一步的說明系統之雙迴圈內部訊號流程。在影像迴圈之中，由駕駛員施力移動力回饋搖桿而產生位移訊號(x)送入飛行虛擬場景，再經由場景端的力感模型計算，而計算出該位移底下需要實際產生多少力量回饋的力量命令(Fr)進而送入力迴圈之中；然而力迴圈主要的目的乃是設計一個力回饋控制器使此迴圈之致動器補償一回饋力(Fm)來與真實力量訊號(Fr)作比較，使回饋力(Fm)能遵照實際力回饋訊號(Fr)的非線性曲線來產生力感。由圖 3.2 上半圖所示[2]，致動器所產生的回饋力(Fm)與場景端的真實力量命令(Fr)首先作比較，比較所產生的誤差命令(Fe)將送入力回饋控制器中，進而去運算出將送入致動器之回饋訊號總值(U)，此時在使用力感裝置的駕駛員便能同時間感受到致動器所給予的回饋力，迴圈最後我們用 Multi I/O 卡來抓取致動器之電壓值並換算出致動器所產生之回饋力(Fm)。因爲此雙迴圈流程互不干擾，所以我們將以兩迴圈分段討論與實作的方式來完成整體飛行模擬架構，在整合系統中屬於影像迴圈的飛行虛擬場景製作與其飛行動態設計我們在將在往後第四章詳細討論，另外，屬於力迴圈之力感裝置設備描述與力回饋控制器的內部設計我們也將在往後第五章中作深入的描述與探討。

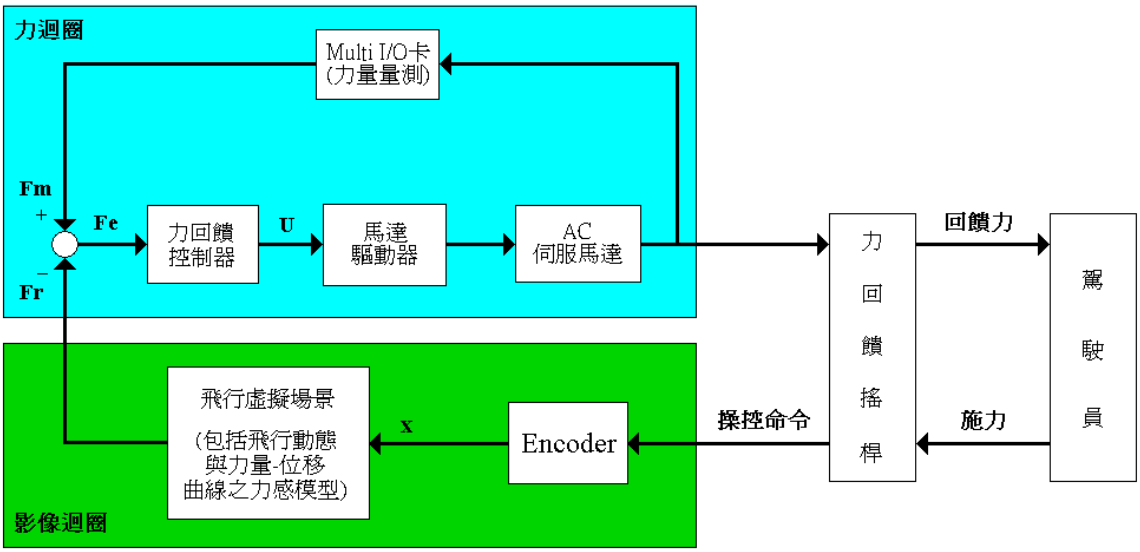

圖 3.2　整合系統之雙迴圈內部訊號流程圖

由於飛行 VR 系統多了力回饋的運算與輸出需要處理，故其系統架構與訊號流程也較爲複雜，因此資訊處理流程也會因爲整合模式的不同而影響到整體飛行模擬系統的效率，故本章第一節便詳細討論雙迴圈之整合方式且比較其優缺點，並說明本研究所採用整合模式之理由。而在第二節中我們將實際撰寫程式，並搭配程式流程圖，已達到軟體實現系統整合的目的。

3.1 雙迴圈整合模式

一般雙迴圈流程的好處是可以依據視覺與觸覺不同的需求分開來處理，在同一部電腦中，可以利用強制性多工 (preemptive multitasking) 的作業系統，如 UNIX、Windows NT 等，採用多行程 (multi-process) 或是多執行緒 (multithreading) 的方法並透過調整行程或執行緒的優先權 (priority) 的方式，依個別需求來合理地分配電腦系統資源。另外，也可以使用跨電腦的方式分別單獨處理一個迴圈以獲取更多的計算能力，例如電腦一 (處理力迴圈) 與電腦二 (處理影像迴圈) 並使用網路 TCP 通訊協定來連結交換相關的資訊。

依據先前雙迴圈之流程並參照圖 3.3 所示，若讀取力感裝置輸入訊號的時間爲 t_i，計算力回饋訊號所用的時間爲 t_{fc}，力回饋輸出的時間爲 t_f，則力更新頻率 f_{r1} 及力的延遲時間 t_{d1} 分別如下：

$$f_{r1} = \frac{1}{t_i + t_{fc} + t_f} \tag{3.1}$$

$$t_{d1} = t_i + t_{fc} + t_f \text{。} \tag{3.2}$$

由於影像迴圈獨立處理飛行動態運算以及力感模型運算，且力迴圈獨立處理力回饋控制運算，其好處是可以分別控制影像迴圈及力迴圈的更新頻率，並提供較高的力更新頻率使模擬系統之力感更逼真。

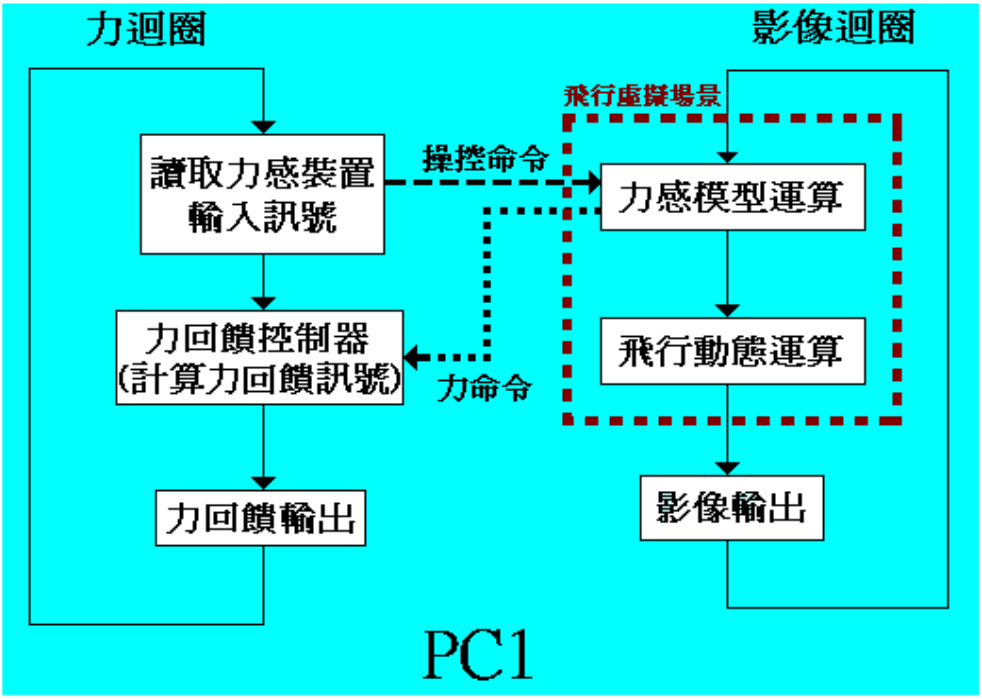

圖 3.3　雙迴圈資訊處理流程圖

然而，圖 3.3 的流程若是在同一台個人電腦上使用雙執行續的方式來處理雙迴圈的話，畢竟會對一台 PC 造成太大的負擔，雖然雙執行續的模式可以提高力迴圈之更新頻率，卻也使得影像迴圈之更新頻率受限於單一電腦的資源而無法維持最佳值[33]。所以我們提出另一個方式來改善這種缺點：以兩台個人電腦分別處理影像迴圈與力迴圈，如圖 3.4 所示，其兩台 PC 之間以區網 TCP 的通訊協定來溝互傳訊號，由於兩台 PC 單獨處理各別的迴圈不但能提供較高的力更新頻率外，負責飛行虛擬場景的 PC 端也能選擇計算 3D 功能強大的繪圖卡與高運算速度的中央處理器專門處理影像迴圈，如此一來，一方面場景端便能提供夠快的資料訊息供力回饋使用，另一方面飛行場景本身更新頻率也能維持住最佳狀態。

依據網路連結雙迴圈之資訊處理流程如圖 3.4 所示，若讀取力感裝置輸入訊號的時間爲 t_i，由力迴圈傳送的操控命令延遲時間爲 t_{ds}，由影像迴圈傳送的力命令延遲時間爲 t_{dr}，計算力回饋訊號所用的時間爲 t_{fc}，力回饋輸出的時間爲 t_f，則力更新頻率 f_{r2} 及力的延遲時間 t_{d2} 分別如下：

$$f_{r2} = \frac{1}{t_i + t_{ds} + t_{fc} + t_{dr} + t_f} \tag{3.3}$$

$$t_{d2} = t_i + t_{ds} + t_{fc} + t_{dr} + t_f \tag{3.4}$$

然而在區域網路之中，通常封包傳遞時間會遠小於 1ms，如此一來 t_{ds}、t_{dr} 便會遠小於 t_i、t_{fc}、t_f，故此時我們便可視 t_{ds} 與 t_{dr} 趨近於 0，則我們可以改寫(3.4)式爲(3.5)式。

$$t_{ds} = t_{dr} \approx 0$$
$$t_{d2} \cong t_i + t_{fc} + t_f \tag{3.5}$$

所以最後我們能得到近似圖 3.3 的效能，而且有優於圖 3.3 的 VR 場景更新頻率，另外由於兩迴圈分開由兩台 PC 獨立處理，所以若有需要更換系統動態與力回饋控制策略則能在短時間之內獨立更新，則此方便性的"模組化"結構更是其系統流程的優點。因此本研究便選擇以網路連結之方式作爲雙迴圈系統整合之架構。

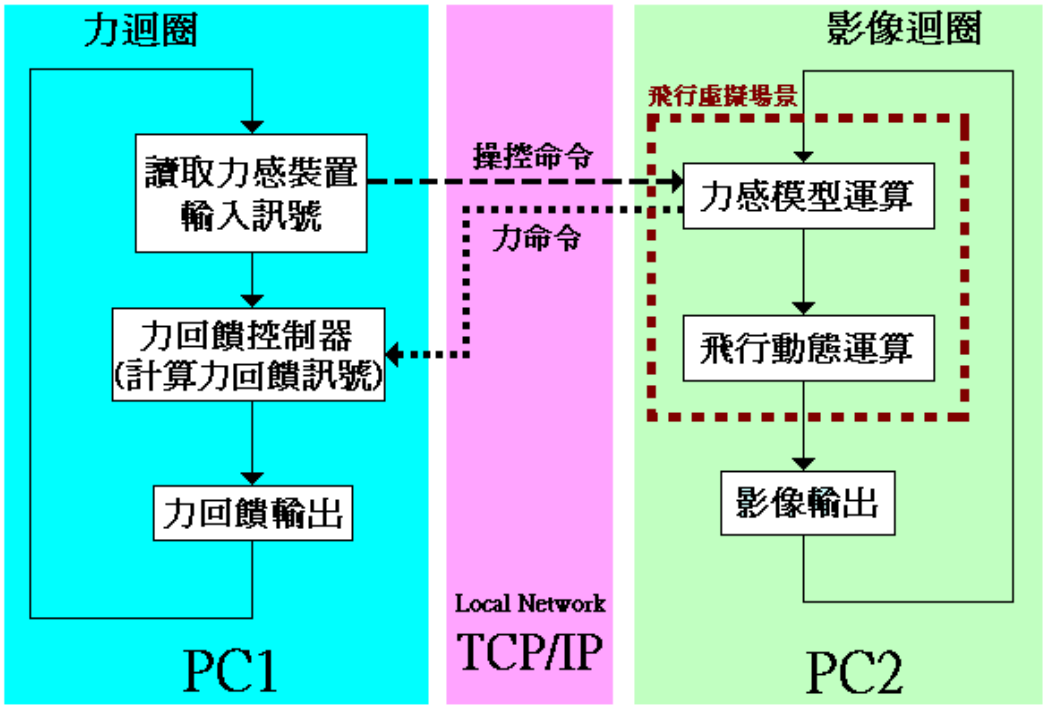

圖 3.4　網路連結雙迴圈資訊處理流程圖

我們將上列的兩種整合模式之優缺點整理如下：

一、其雙迴圈係由單部電腦採用雙執行續方式處理，如圖 3.3 所示，本方法的優點是可以分別控制影像及力的更新頻率，並提供較高的力更新頻率，但缺點是造成單部電腦運算量過大，且影像迴圈之更新頻率無法維持最佳狀態。

二、使用兩部電腦分別處理雙迴圈，並經由區域網路來傳遞訊號，如圖 3.4 所示，本方法的優點同一，且模組化的架構便於更改其影像迴圈的系統動態與力迴圈的控制策略，此架構甚至於能使力感系統相容於不同的 VR 引擎與系統平臺，擁有相容性及高的優勢；但缺點除了流程及方法較複雜外，並受限於網路距離與速度，若是網路連線距離過長而導致於網路速度過慢且延遲，則會使得系統效能變差。

3.2 網路整合之實踐流程

而在開發環境的作業系統選擇上，一方面因爲我們所使用的馬達控制卡和 Multi-I/O 卡的驅動程式，它們所支援的作業系統有限，另一方面也因爲我們選擇使用兩部電腦分別處理雙迴圈，並經由區域網路來傳遞訊號，基於這兩項原因，所以我們選擇微軟 Windows 2000 做爲力回饋搖桿之控制電腦所發展的作業平台，而網路架構我們選擇針對 Windows TCP/IP 通訊協定所設計的 WinSock 介面程式來開發，因爲 WinSock 介面於 NT 平台架構的 Windows 2000 底下擁有傳輸快速、穩定與高相容性的優點。故本研究所用到的發展程式語言工具爲支援以上硬體與軟體需求的微軟 Visual C++ 6.0，但不以 MFC (Microsoft Foundation Classes) 的方式來撰寫程式，原因在於，雖然 MFC 提供了方便的微軟視窗應用式發展環境，但它的程式架構並不符合我們想要設計的程式流程，而且 MFC 無法搭配其他的電腦繪圖軟體使用，再加上 MFC 只能在微軟的作業系統上使用，並不利於將來可能需要的跨平台發展。

在程式設計部分，我們撰寫 WinSock 網路程式[40]來連結雙迴圈中的力以及影像資訊，而針對上一節中網路連結雙迴圈資訊處理流程，我們將其規劃成圖 3.5 所示的程式流程圖，其系統動態可置於 VR 影像迴圈單獨來處理。另外我們並配合完整的 client-server model 程式設計流程圖來幫助程式撰寫，其資料如圖 3.6 所示，為了方便解說，我們將流程圖分為 Server 端與 Client 端分別說明。

◆ **WinSock Server 端(即為 VR 影像端)流程：**

1. Server 端首先開啓 WSAStartup( ) 函式，其功用是讓所有應用程式與 WinSock.DLL 連結，故每個應用程式都必須先執行此函式。
2. 呼叫 socket( ) 函式去開啓一個 socket 作為將來資料傳送的管道。
3. 呼叫 bind( ) 函式告訴 WinSock 程式指定 Server 端的 IP 與 port。
4. 呼叫 listen( ) 函式使此 socket 進入「監聽」狀態，並開始接受 client 的連線要求。
5. 利用 accept( ) 函式接受由 Client 端來的連線要求，並建立連線。
6. 使用 recv( ) 函式接收來自 Client 端所傳送的位置命令(x) (必須搭配 Client 端的 send( ) 函式一起使用)。
7. 由 Client 端所收到的位置命令(x)直接送進 VR 影像迴圈之中處理，其中 INITIAL_SCENE( ) 副程式功能為初始化整個 VR 繪圖環境，SCENE_GRAPH( ) 副程式功能為繪製場景架構，其內部流程將會在往後場景繪製章節中詳細解說。而最後位置命令(x)將會送進系統的動態區塊與力感曲線模型區塊之中分別計算出系統應有的特徵值與力量命令(Fr)。
8. 使用 send( ) 函式發送力量命令(Fr)至 Client 端作力回饋控制 (必須搭配 Client 端的 recv( ) 函式一起使用)。
9. 呼叫 Closesocket( ) 函式將開啓的 socket 關掉。
10. 最後呼叫 WSACleanup( ) 函式釋放所佔用的 Winsock.DLL 資源。

◆ **WinSock Client 端(即爲力回饋控制端)流程：**

1. Server 端首先開啓 WSAStartup( ) 函式，其功用是讓所有應用程式與 WinSock.DLL 連結，故每個應用程式都必須先執行此函式。
2. 呼叫 socket( ) 函式去開啓一個 socket 作爲將來資料傳送的管道。
3. 呼叫 connect( ) 函式向 Server 端發出連線要求，如果成功了表示 Server 端接受連線要求，可以準備開始傳送資料。
4. 由此刻便開始進入力迴圈，首先會進行力迴圈環境初始化的設定，其中包括宣告力搖桿位置在內的全域變數 (global variable)、及力回饋搖桿驅動程式的初始化，並可經由 NCC9322 馬達控制卡抓取實際搖桿之位移值(x)。
5. 使用 send( ) 函式發送位置命令(x)至 Server 端作系統動態的更新與應有實際力回饋值的計算(必須搭配 Server 端的 recv( ) 函式一起使用)。
6. 使用 recv( ) 函式接收來自 Server 端所傳送的力量命令(Fr)(必須搭配 Server 端的 send( ) 函式一起使用)。
7. 將由 Server 端所接收的力量命令(Fr)送入力回饋控制器之中作運算，並驅動馬達輸出實際力搖桿之力感，而有關力回饋控制器結構我們將於第五章作詳細說明。
8. 呼叫 Closesocket( ) 函式將開啓的 socket 關掉。
9. 最後呼叫 WSACleanup( ) 函式釋放所佔用的 Winsock.DLL 資源。

以上兩個流程便是此研究之程式流程重點所在，而此兩個迴圈皆以不同的頻率於兩台電腦上執行，通常力迴圈更新率比較高，而 VR 影像迴圈更新率比較低。由於整體系統是經由網路整合，故 VR 影像迴圈更新速率通常僅受限於場景端本身資源，而力迴圈卻需要強制釋放適當比例系統資源以搭配 VR 影像迴圈來使整體表現性更佳，如使用 Sleep 指令等。

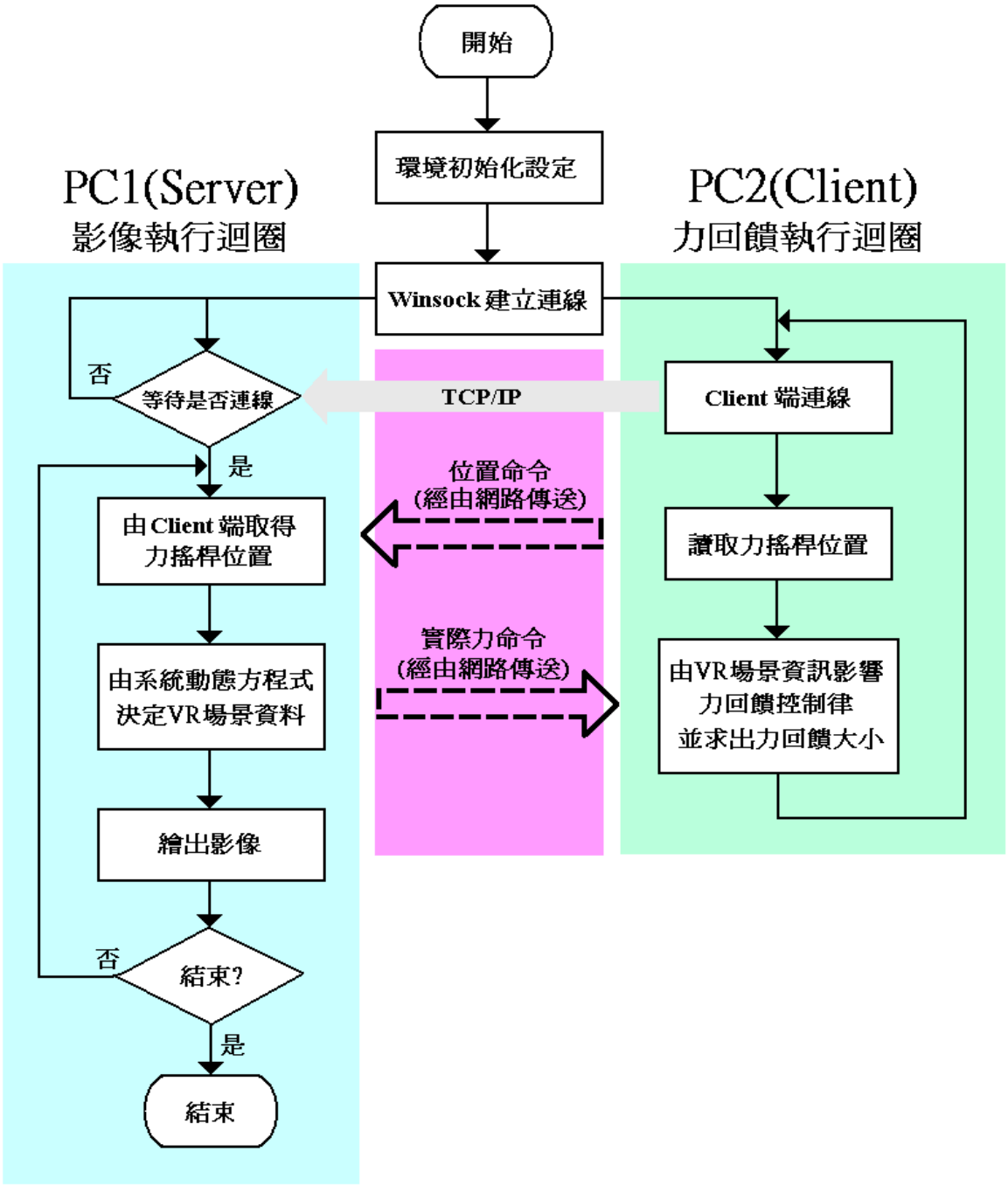

圖 3.5　雙迴圈之程式流程圖

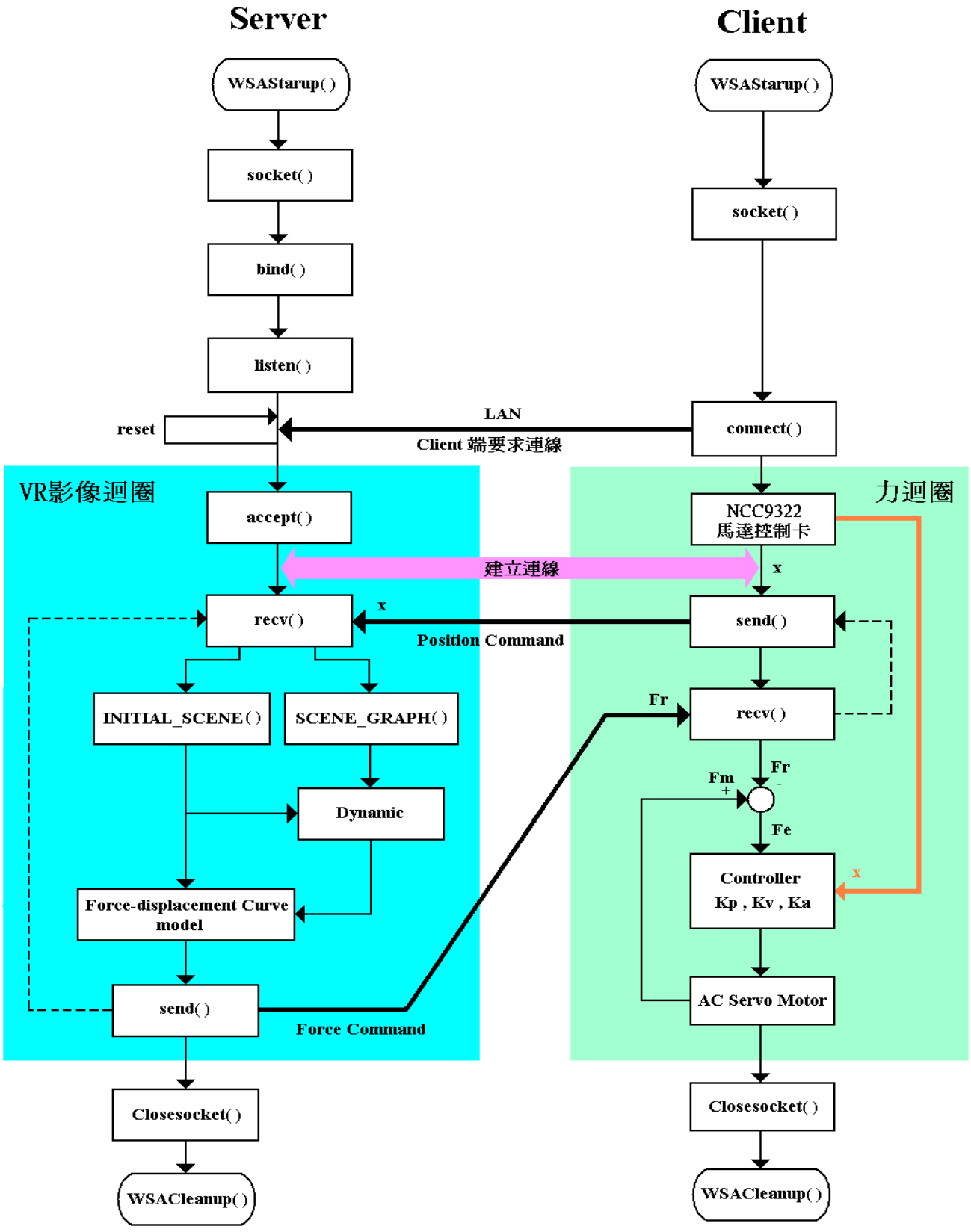

圖 3.6　雙迴圈連接導向(TCP)的 client-server model