國 立 交 通 大 學

電 信 工 程 學 系 碩 士 班

碩 士 論 文

分散式即時互動Client/Server影像辨識系統

Real-Time Interactive Client/Server Distributed Image Recognition System

研 究 生：陳其瑩

指導教授：張文鐘 博士

中 華 民 國 九 十 四 年 八 月

分散式即時互動Client/Server影像辨識系統

Real-Time Interactive Client/Server Distributed Image Recognition System

研 究 生：陳其瑩　　Student：Chi-Ying Chen

指導教授：張文鐘 博士　Advisor：Dr.Wen-Thong Chang

國 立 交 通 大 學

電 信 工 程 學 系

碩 士 論 文

A Thesis

Submitted to Department of Communication Engineering

College of Electrical Engineering and Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Communication Engineering

August 2005

Hsinchu, Taiwan, Republic of China

中 華 民 國 九十四 年 八 月

分散式即時互動 Client/Server 影像辨識系統

研究生：陳其瑩　　指導教授：張文鐘 博士

國立交通大學電信工程學系碩士班

摘　　要

近年由於電腦科技的進步，使得視訊監控系統廣泛的佈建在生活周遭。然而在現今的監控系統平台架構下，因系統的運作機制並非相當完善，如欲達到監控的目的，需要大量的人力及時間。以目前的系統架構而言，取像設備、監控中心、監控人員三者大都位於同一區域，也就是說監控人員需要在監控中心存取影像檔案。另外為了判斷是否有異常現象發生，監控人員需要不間斷的觀看相機影像，當有事件發生時，再以人工設定的方式進行錄影的動作。另外當系統提供多個相機影像時，觀看相機的選取需透過人工設定的方式達成。由上可知整個系統架構所提供的人機互動性並不高，往往需要人工設定才能執行相關的動作。

有鑑於監控平台運作方式的不完善，因此我們提出即時互動式監控系統來加以解決。在即時互動式監控系統架構下，存在著 Client/Server 的概念，Server 端負責影像資料的收集，監控人員可透過 Client 端的程式向 Server 端存取監控影像，達到遠端檔案存取的功能。另外因為我們在 Client 端加入物件定位及手勢辨識等影像處理，使 Client 可對監控影像進行影像處理的動作，進而將運算結果回報給 Server 端，Server 端再依 Client 端的回報訊息進行相對應的處理。Client/Server 的分散式架構搭配相關的影像處理，使系統達到自動錄影機制、自動相機切換機制、遠端檔案存取功能。

Real-Time Interactive Client/Server Distributed Image Recognition System

Student：Chi-Ying Chen Advisor：Dr.Wen-Thong Chang

Department of Communication Engineering

National Chiao Tung University

Abstract

Due to the dramatic advances of the computer technology, surveillance systems are widely accepted and used. However, a huge amount of human resources and time are required to monitor these systems due to the lack of a completely operational mechanism. Specifically, for the current model, capture devices, the surveillance center, and the surveillance personnel have to be at the same location to accomplish the task. In other words, the surveillance personnel must always stay in the surveillance center to look at the monitors closely in order to see whether an abnormal event occurs or not. When an abnormal event occurs, the surveillance personnel have to manually turn on the recording devices. If multiple cameras are present, the surveillance personnel also have to manually switch between different cameras. From the above, it is not hard to conclude that the level of human-machine interaction for this model is not high enough. It requires lots of manual settings.

To overcome this major drawback, in this paper, a real-time interactive surveillance system is proposed. In this model, the client/server concept is put in practice. The server system is responsible for collecting visual data from various types of cameras. Thus, the surveillance personnel can remotely acquire these collected data with the use of the client system. Moreover, because the object positioning and the gesture recognition techniques are added to the client system, the client system is able to process the acquired image data. When detecting an abnormal event, the client can inform the server about it. The server in turn can react to the event according to the pre-defined rules. The proposed distributed client/server architectures in conjunction with the image processing capabilities provide the surveillance system with the abilities to automatically record, switch between cameras, and access files remotely.

誌謝

首先，我要感謝我的指導教授張文鐘老師，感謝老師兩年來辛苦的指導，以及提供實驗室良好的研究資源，使學生可以順利的進行多媒體通訊領域的相關研究。

其次，感謝實驗室中的學長，在研究方面給予我莫大的幫助及教導，你們的知識傳承使我受益良多。接著我要感謝一起打拼的實驗室夥伴，包括建華、智維、義浩、心賢、旃偉等，感謝你們在課業及生活上的協助。

最後，感謝我的家人及女朋友阿芳，不斷地給予我關懷與支持，成為我讀書的原動力，使我能順利地完成學業。

目 錄

圖目錄

表目錄

第一章 簡介

1.1 研究動機

近年由於電腦科技的進步，使得視訊監控系統廣泛的佈建在日常生活周遭，相關的影像處理應用也與日俱增。諸如現今的車牌辨識、指紋辨識、文字辨識、入侵者偵測等等，都是利用影像處理技術所實現的。而在視訊監控的範疇裡，移動物體的定位與追踪是很重要的應用，可應用在監控系統、居家看護等之應用。除了上述所說的物件定位與追踪之外，手勢辨識也是監控系統裡重要的應用，藉由手勢的動作，使系統自動辨別其手勢訊號，並根據手勢訊號執行相關動作。

在監控平台互動性方面，現今的系統架構所提供的人機互動性並不是很高，諸如相機的切換機制、錄影機制、檔案存取方式等等，往往需要人們總總的設定才能執行相關的動作，因此人機互動機制的提昇是必需的。

1.2 相關研究

在監控平台方面，以目前的監控系統架構而言，取像設備、監控中心、監控人員三者大都位於同一區域，也就是說監控人員需要在監控中心存取影像檔案，並不支援遠端檔案存取的功能。另外，監控人員為了

要判別有無事件的發生，需要時時刻刻的觀看每一相機的影像畫面，當有事件發生時要立刻處理或執行錄影的動作。上述影像檔案的存取方式及人機互動的不足造成大量的人力消耗。

針對移動物件定位與追踪的技術方面，目前大都是應用在單一移動式(例如PTZ camera)的取像設備架構下所執行的，也就是說只針對一個取像設備所擷取到的影像進行分析。在此系統架構下，如要達到物件定位與追踪的目的，所需執行的運算流程有影像擷取、物件抽取、移動估測等等。經由此運算流程之後，可得物件的座標及移動方向，有了此資訊之後，便可調整相機的參數，使其鏡頭移動在可定位物件的位置。以上是一般在進行物件追踪時，所使用的系統架構及運算流程。

在手勢辨識方面，早期是利用戴手套的方式來進行手勢辨識。手套上有特殊的光點分佈，藉由光點比對的方式，進行手勢辨識的運算。後來，有人利用手部的幾何分佈、色彩特徵來進行手勢辨識的工作。另外還有人利用隱藏式馬可夫模型(Hidden Markov Model)，建立手語模型的方式進行手勢辨識的處理。此外，有人利用Modified Fourier Descriptors的方式將各手勢模型分類，進行手勢辨識的工作。最後，有人利用手部特徵點的分佈，進行手勢辨識。

1.3 論文架構

在第一章裡，將介紹目前的視訊監控系統架構及運作方式，以及在單一移動式相機架構下所發展的物件定位與追踪技術。另外，回顧一些過去及近年手勢辨識的運算方式。在第二章裡，介紹我們提出的即時互動式監控系統所使用的即時傳輸技術。在第三章裡，將提出在即時互動式監控系統下，是如何的透過 RTSP 協定達到人機互動的功能，進一步介紹所提供的互動機制。在第四章中，將介紹一種在雙相機系統架構下，物件的定位技術，及在此系統架構下，相機的換手機制。在第五章中，我們提出一種利用手部特徵點分佈的資訊，進行手勢辨識的運算。在第六章中，將介紹監控平台的系統實現部分，依序介紹 Client/Server 端的使用者介面、Client 端程式運作流程。第七章則是結論部分。

第二章 即時傳輸技術

2.1 簡介

在傳統的資料傳輸架構下，皆是使用 TCP/UDP 協定進行傳輸，然而如單單使用 TCP/UDP 進行多媒體資料傳輸時，並不能達到即時傳輸的要求。為了要達到多媒體即時傳輸的目的，我們通常搭配 RTP(Real-Time Transport Protocol)及 RTCP(Real-Time Control Protocol) 進行即時檔案傳輸。其中 RTP 是針對即時傳輸系統提供較合適的資料傳輸。因其 RTP header 裡包含了 TimeStamp 的資訊，使時間的觀念引入了此系統，使得接收端有更多的資訊去處理 received date。然而在傳輸資料時，因網路特性等因素，必有傳輸流量等課題需考量，因此我們藉由 RTCP 協定，取得 Receiver 端的資料接收情形，進而回報此資訊給 Sender 端，使 Sender 端適度的調節傳輸速率。

2.2 RTP

2.2.1 RTP 簡介

傳統的資料傳輸都是建構在 TCP/IP 網路架構下所完成的，也就是說在各網路間流通的資料為 IP packet。而 IP packet 裡的 TCP header 裡只有 SequenceNum 的資訊，因此用戶端只能利用 SequenceNum 的資訊

重建各 packet 之間的順序。然而 SequenceNum 資訊只解決各 packet 的重建順序，卻無法提供即時傳輸的要求。要解決即時傳輸要求的話，我們可搭配RTP協定達到即時傳輸的要求。因為RTP header裡有 timestamp 的欄位，此欄位記錄著各影像、聲音 packet 所應播放的時間。藉由 SequenceNum 及 timestamp 的資訊可增進即時傳輸的品質。

2.2.2 RTP Packet Format

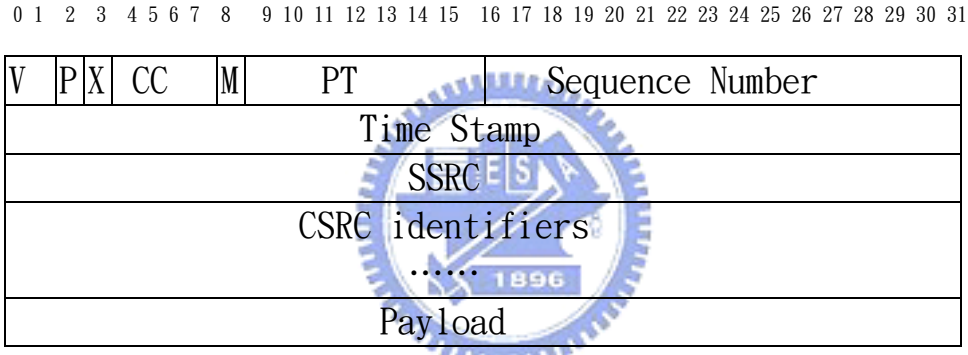


圖 2-2-1 RTP Packet Format

圖 2-1-1 為 RTP Packet Format[1]，各欄位的名稱及 bit 數如下：

V：version (2 bits)

P：padding flag (1 bit)

X：header extension flag (1 bit)

CC：CSRC count (4 bits)

M：marker (1 bit)

PT：payload type (7 bits)

Sequence number (16 bits)

Timestamp (32 bits)

SSRC：synchronization source (32 bits)

CSRC：contributing source (0~15 items, 32 bits each)

Payload(32)

我們由 RTP Packet 的各欄位裡，說明較重要的欄位意義：

Sequence Numbers：

此欄位記載著封包傳輸時的順序，使接收端能依此資訊重整封包的順序，進而重建檔案。另外，此欄位資訊對我們在計算packet loss、進行packet duplication幫助很大。

Timestamp：

此欄位記載著時間的訊息，使接收端能利用此資訊，計算 jitter 及資料同步的功用。

SSRC：

此欄位主要功用是使接收端能判別每一個 RTP packet 的 Sender 資訊，作法為針對每一個 RTP Session 給予一個獨一無二的 random number，因此不同的 session 會有不同的 SSRC，因此接收端可輕易的知道每一 RTP packet 由哪一個 Sender 所傳送的。

CSRC：

假設網路上有二個 Sender，二者對應的 Receiver 都是同一個，因

此存在二個 RTP session。此時因為某種應用的原因，使這二個 RTP session 所傳送的封包透過一 Mixer 將此二封包組成一個新的 RTP packet，因此新的 RTP packet 需要利用 CSRC 記錄此 packet 是由誰所組成的。以下列子說明 CSRC 的功用：

SSRC & CSRC Example：

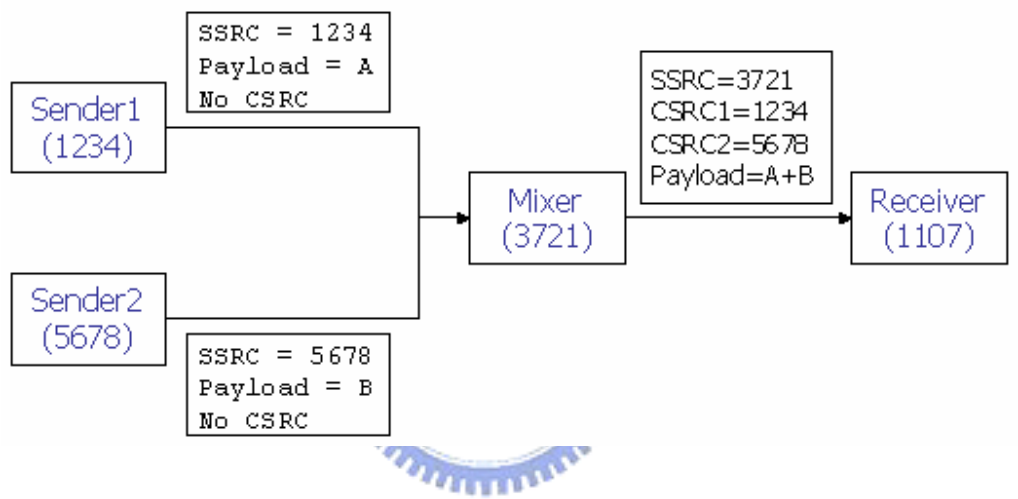


圖 2-2-2 SSRC & CSRC Example

藉由上圖的例子，說明 SSRC 及 CSRC 二欄位在即時傳輸時的作用。在 SSRC 方面，Sender1、Sender2 可以各自指定一個 random number 為 SSRC，將此 SSRC 設定在 RTP Header 裡，使 Receiver 可分辨某一時間的 incoming packet 是 Sender1 或是 Sender2 送來的。

假設網路上有一 intermediate system(ex:Mixer)，此時 Sender1、Sender2 的 packet 可能會透過 Mixer 重組成一個新的 packet，因此這個 Mixer 的 output packet 必須清楚的記載著此新的 packet 是由何者

提供的，如上圖，CSRC1=Sender1’s SSRC、CSRC2=Sender2’s SSRC，因此當 Receiver 接收到此 packet 時，可由 CSRC 欄位的資訊知道此 packet 由那些 sender 組成。

2.3 RTCP

2.3.1 RTCP 簡介

當我們使用 RTP 方式傳送 data 時，Sender 端只負責傳輸資料，並不在乎 Receiver 端接收資料的情形，此時如 Receiver 端忙錄或接收資料情形不理想的話，可能會造成大量的 packet loss。為了使 Sender 端在傳送 data 時能考慮 Receiver 端的接收情形，我們常常使用 RTCP protocol 來得知 receiver 端接收資料的情形，進而回報給 Sender 端，以適度的調節傳輸速率。以下我們將介紹 RTCP 的 Packet types，及它到底是如何利用不同的 packet type 得知 receiver 端的接收情形，並回報給 sender 端。

2.3.2 RTCP Packet Format

RTCP Packet 可分成 5 種型式[2]，列舉如下：

SR : Sender report

RR : Receiver report

SDES : Source description items, including CNAME

BYE : Indicates end of participation

APP : Application specific functions

在上述 5 種 packet type 中，以 Receiver report 的 packet type 最重要，因為它是眾多的 RTCP packet 裡，記載最多關於 Receiver 接收資訊的訊息。以下為 Receiver Report Packet 的 packet format：

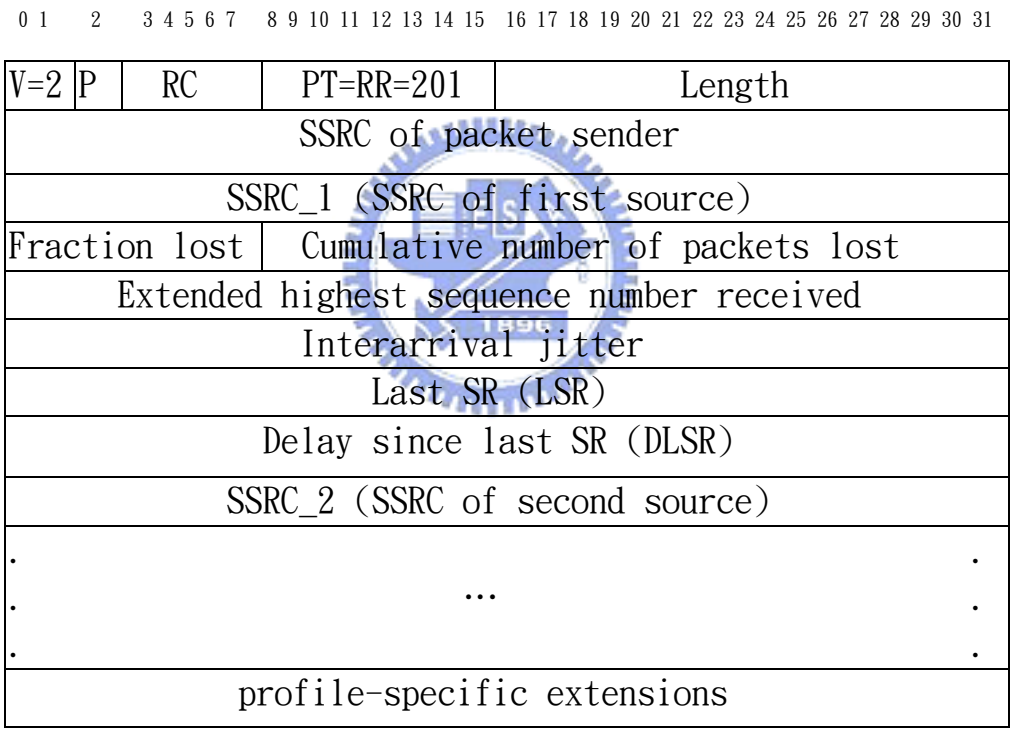


圖 2-3-1 Receiver Report Packet Format

圖 2-3-1 為 Receiver Report Packet Format，以下將介紹此 Packet Format 中較重要的欄位及其意義：

Fraction lost (FL)：

記載著 packet loss 的機率，以 0~255 來表示。Receiver 端以某一時間區間為單位，累加其 packet loss 數，可得 probability of packet loss=num of packet loss/num of received packet，再將此值 scale 到 0~255。

Inter-arrival jitter：

計算二個連續的 packet 到達 receiver 時間的 variance(jitter)以下列數學式計算得知：

$$J = J + \frac{(|(R_n - R_{n-1}) - (S_n - S_{n-1})| - J)}{16} \qquad (2.1)$$

其中 J 為 inter-arrival jitter，R_n 為 arrival time，S_n 為 send time。

LSR、DLSR：

LSR 欄位記載 sender 端傳送此 packet 的時間(NTP Timestamp)，DLSR 記載 receiver 端處理某一 packet 所花的時間，LSR、DLSR 二參數都是為了要用來計算 RTT，以下面例子說明：

Round-Trip Time(RTT) Calculation：

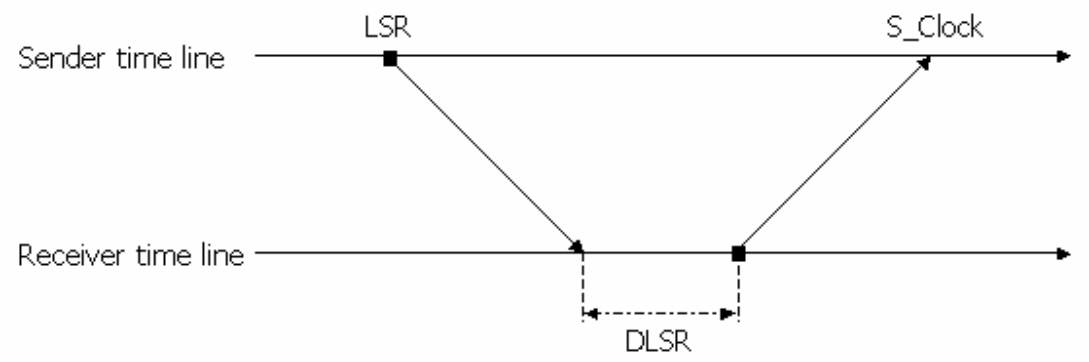


圖 2-3-2 Round-Trip Time 計算

起初利用 LSR 記載 sender 端傳送此 packet 的時間，接著經過 channel 到 receiver，此時 receiver 會計算其對此 packet 的處理時間(DLSR)，然後回報訊息給 sender。當 sender 收到此 report 時，記錄系統時間(S_Clock)，再利用 report 內的相關參數(ex:LSR、DLSR)及自身的 S_Clock 訊息，可算出其 RTT 。

$$RTT = (S\_Clock - LSR) - DLSR \qquad (2.2)$$

由上述參數的介紹之後，可知 sender 端可藉由 Receiver Report Packet 內的變數 LSR、DLSR 算出 RTT。也可由 Fraction lost 知 packet loss 的情形。另外由 Inter-arrival jitter 算出封包 arrival time 的 variance。因此 sender 端經由 receiver 回報的 Receiver Report Packet 可知 receiver 針對此連線的資料接收情形，接著再配合其他 packet

type 的封包，可完整得知 reception quality 的資訊，進而調整傳送資料的速率以符合 receiver 端的接收情形。

第三章 互動式監控系統

3.1 簡介

在傳統的監控系統裡，為了達到監控的目的，大都是以人工的方式觀看取像設備所擷取到的影像檔內容，逐一觀看每一時刻的影像畫面，再判別是否有事件發生。然而在此系統架構下，很多的功能都需要人們手動去設定才行。諸如相機的切換 、錄影動作等等，都需要人們手動設定才行，人機互動性的不足造成大量的人力消耗。有鑑於傳統監控系統的人機互動性不足，我們提出一種具互動性質的視訊監控系統，藉由RTSP(Real Time Streaming Protocol)所提供的遠端控制功能，以及我們所實現的影像處理程式，使其達到人機互動的功能，諸如相機的換手機制、錄影機制、遠端即時檔案存取、事件分析與處理等等，都可藉由我們所佈建的監控平台予以實現。以下我們將介紹為了要達到此互動性功能所需搭配使用的RTSP協定，以及搭配RTSP Method所完成的互動機制。

3.2 RTSP

3.2.1 RTSP 簡介

RTSP(Real Time Streaming Protocol)是一種在即時檔案傳輸環境下，控制檔案傳輸的協定，以 OSI 七層架構來說，是屬於應用層的協定。然而在眾多 Streaming 相關的 protocol 裡就屬 RTSP protocol 最重要，它扮演著 client、server 二端的訊息溝通，使得此 Streaming procedure 得以運作正常。換句話說，RTSP 提供一種即時檔案傳輸時可控制的架構，使在進行多媒體服務時，可達到網路遠端控制的功能。

3.2.2 Procedures of RTSP connection

在此將介紹 Client、Server 二端是如何建立 RTSP connection，及建立連線的過程中所經的程序，以圖 3-2-1 說明[3]、[4]：

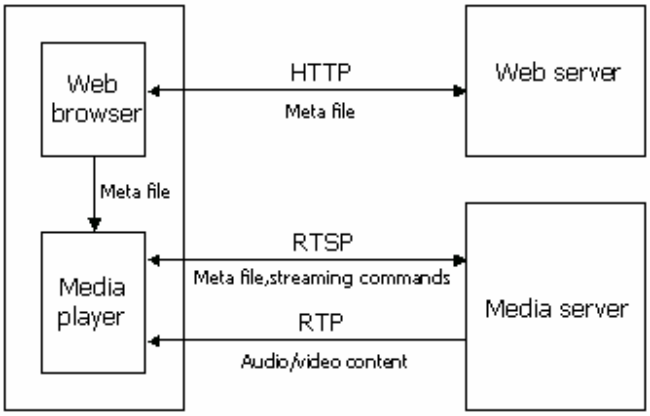


圖 3-2-1 procedures of RTSP connection

由圖 3-2-1 可知 RTSP connection 的建立流程，首先 Client 端先

利用 HTTP protocol(GET)和 web server 接觸，接著 server 端回送 Client 一個 meta file，此檔案記載著 Media Server 的相關訊息。完成上述程序之後，Client 端的 Media player 便可利用 RTSP method 和 Media Server 進行控制訊息的溝通。爾後，便透過 RTP 進行 Multimedia Data 的傳送。

3.2.3 RTSP Methods

RTSP 是一種在 Client、Server 二端進行即時檔案傳輸時，所使用的控制協定，Client、Server 二端可利用 RTSP 協定所提供的 Method 進行溝通。以下列出 RTSP Method，並介紹各 Method 的作用[3]：

表 3-2-1 RTSP Method Definition

Method	Description
DESCRIBE	取得 Media Data 的相關描述
GET_PARAMETER	取得 URL 中描述 stream 的相關參數
OPTION	取得此 RTSP Session 可使用的 Methods
PAUSE	執行暫停的動作
PING	Client、Server 用來檢查對方是否處於連線狀態
PLAY	執行播放的動作
REDIRECT	告知 client 端必須連結到另一個指定的 server

SETUP	建立 RTSP Session
SET_PARAMETER	設定 URL 中描述 stream 的相關參數
TEARDOWN	關閉此 Session，並釋放相關資源

3.2.4 RTSP Operations

經由章節 3.2.2、3.2.3 的介紹之後，對 RTSP Connection 的建立以及 RTSP Method 的使用時機有了初步的了解。然而一個完整的 RTSP Connection 需要透過很多運算及 RTSP Method 的使用才能完成，以下圖說明[4]：

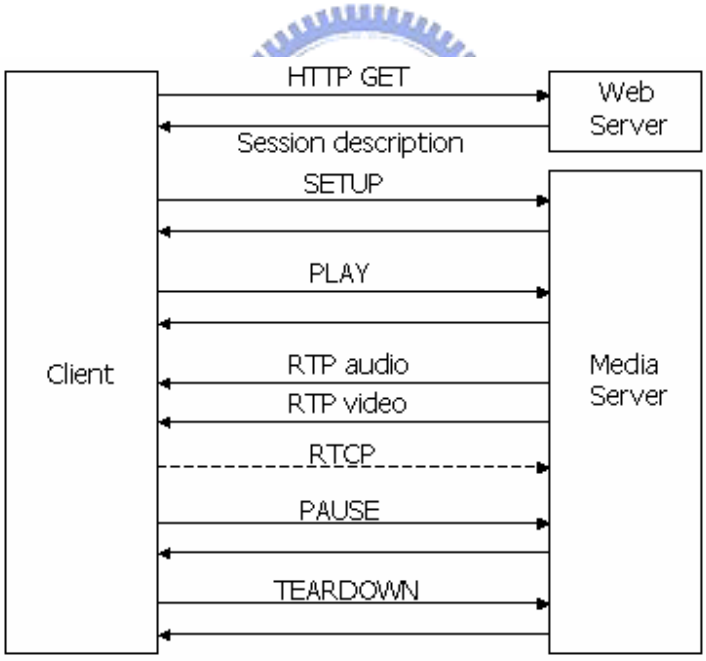


圖 3-2-2 RTSP Operations

藉由圖 3-2-2 的說明，我們可更進一步了解 Client/Server 如何利用 RTSP 所提供的 control message 達成資料的傳遞。起初仍是利用 HTTP

protocol(GET)和 web server 接觸，接著 server 端回送 Client 一個 Session Description 訊息，此訊息記載著 media server 的相關資訊。經上述程序之後，Client/Server 二端便可利用 RTSP Method 進行控制訊息傳遞。首先，Client 送出一個 SETUP Method 給 media server，假設 server 端同意此請求的話，它會送出去一個相對應的 response 以回覆此 request。完成上述設定之後，表示此 RTSP Connection 已完成，Client 便可送出 PLAY Method 請求播放影音檔案。同樣的，server 端會送出去一個相對應的 response 以回覆此 request，前述所說的流程都是在傳送 Streaming data 前的設定動作。接著 server 端便可將 media data 封裝成 RTP Video、RTP Audio packet 傳送給 Client 端。然而在傳輸資料時，因網路特性等因素，必有傳輸流量等課題需考量，因此利用 RTCP protocol 回報 receiver 端的接收情形，使 Server 端適度的調節傳輸速率。經過一段時間後，假設 client 端想暫停影片播放，便可發送 PAUSE Method 請求暫停或是利用 TEARDOWN Method 關閉此 Streaming data 連線。

3.2.5 RTSP State Machine

一般在建立 Streaming 連線時，皆是使用 Client/Server based 的架構，而每個(Client、Server) pair 為一個 Session。但通常一個 Server

不可能只服務一個 Client，常是一個 Server 提供服務給多個 Client，因此會建立很多 Session。如圖 3-2-3 所示：

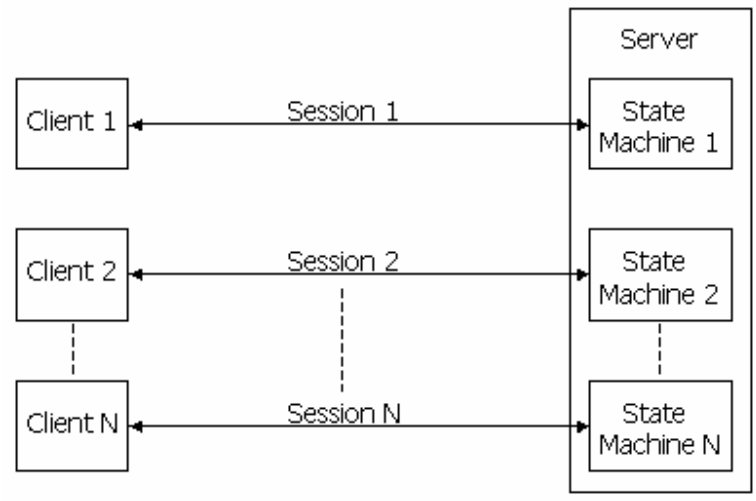


圖 3-2-3 RTSP Session

在 Server 端需面對多個 Client 時，如何清楚的記錄每個 Session 的狀態便顯得格外重要。此時 State machine 的觀念替我們解決了上述問題。如圖 3-2-3 所示，Server 端針對每個 Session 建立個別的 State machine，利用 State machine 記錄每個 Session 的狀態，以利處理每個 Client 的要求及其本身所應回覆的相對動作。以下說明 RTSP State Machine 的運作機制，以圖 3-2-4 說明：

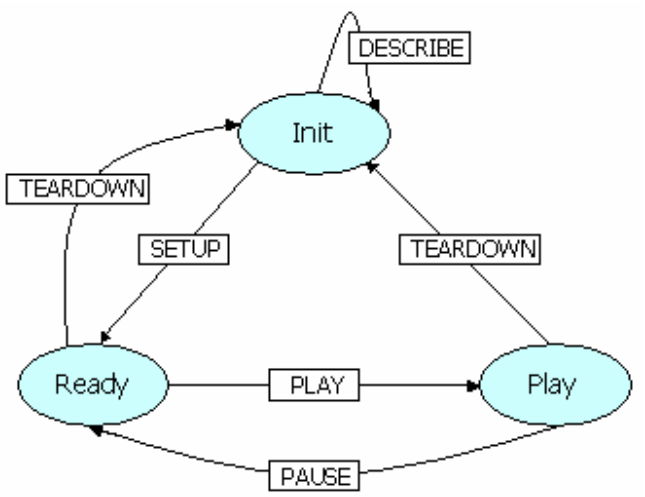


圖 3-2-4 RTSP State Machine

圖 3-2-4 為 RTSP State Machine 的運作流程圖[3]，在 RTSP 的相關應用下，我們將 States 分為 3 種，分別為 Init_state、Ready_state、Play_state。各 state 間會因 Input signal 的不同而產生不同的 State Transition。舉例來說，假設現在是處於 Init_state，如果 Input Method 為 SETUP，則會產生 State Transition 的情形，會由原本的 Init_state 跳到 Ready_state。因此 Server 端可依 State Machine 的狀態決定該提供什麼樣的服務給對應的 Client 端。

3.3 互動式平台系統架構

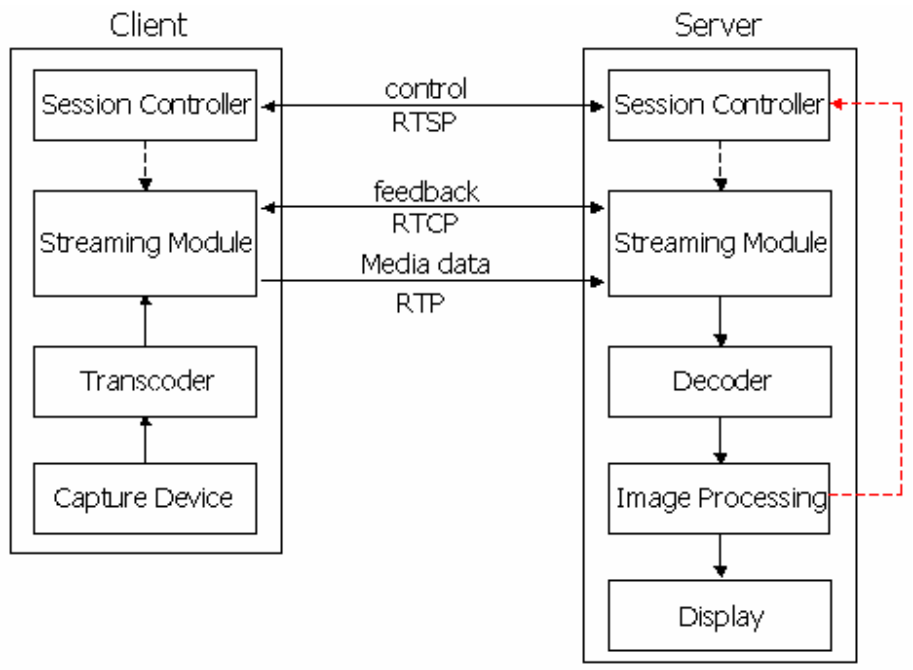


圖 3-3-1 互動式平台

圖 3-3-1 為互動式平台的系統架構，整個系統主要可分為二部分，分別為 Server 端及 Client 端。Server 端的主要任務為影像資料的取得，及將取得的影像資料利用串流的相關技術將資料傳輸到 Client 端。目前所使用的取像設備為固定式的 IP camera，影像的輸出格式為 JPEG，但因為 Server 端最後要傳送出去的格式為 MPEG4 格式，因此 Server 需將從 IP camera 取得的影像進行 trans-coding 的動作，最後再透過 Streaming module 將 MPEG4 資料傳輸出去。

在資料傳輸部分，多媒體資料的即時傳輸是使用 RTP 協定來完成，並搭配 RTCP 協定控制資料的傳輸速率。另外 Client、Server 二端藉由 RTSP 協定達到互相溝通的動作。

Client 端的主要工作為將接收到的 Streaming data 透過 Decoder 還原成 RGB 的影像型式，接著再將此影像資料交給 Image Processing 程式進行相關的影像處理，最後再依影像處理後的結果來控制 Client 端的 Session Controller，Session Controller 會依各事件定義發出相對應的 RTSP Message 給 Client 端，使 Client、Server 二端達到互動的功能。

3.4 互動機制

在章節 3.3 中，介紹了互動式平台的系統架構。然而在此平台下，我們定義了下列的互動機制，包含遠端即時檔案存取、相機之換手機制、自動錄影機制。

[遠端即時檔案存取]

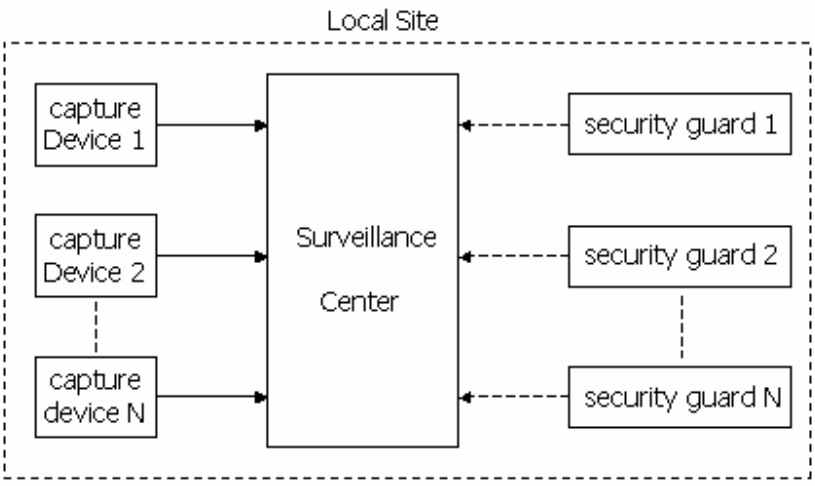


圖 3-4-1 傳統式檔案存取

圖 3-4-1 為在傳統的監控系統架構下，影像檔案存取的方式。如圖所示，監控中心藉由各取像設備收集影像資料，安管人員如要觀看監控的畫面時，需要直接向監控中心存取影像資料。在此架構下，取像設備、監控中心、安管人員三者都位於同一場所。

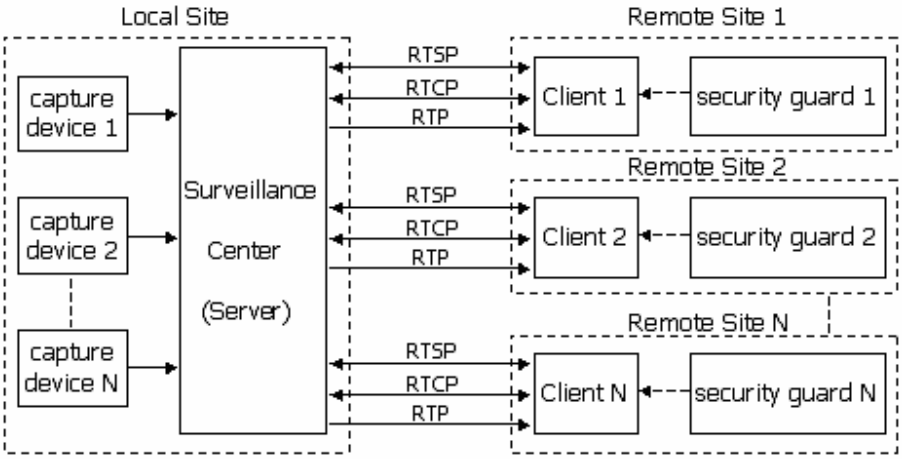


圖 3-4-2 遠端即時檔案存取

然而在我們所佈建的監控平台下，影像檔案的存取方式變的更有彈性，存取方式如圖 3-4-2 所示。取像設備和監控中心位於 Local site，安管人員可在 Remote site 執行 Client 端程式，此程式會利用 RTSP 協定和監控中心(Server)溝通，並透過 RTP、RTCP 處理即時的檔案傳輸，使安管人員可從 Remote site 存取 local site 的影像資料，達到遠端存取的功能。

[相機之換手機制]

一般的監控系統架構下，如監控中心掌管多台取像設備時，此時安管人員需以人工方式設定監控中心 Displayer 的顯示畫面，上述相機切換機制以圖 3-4-3 呈現：

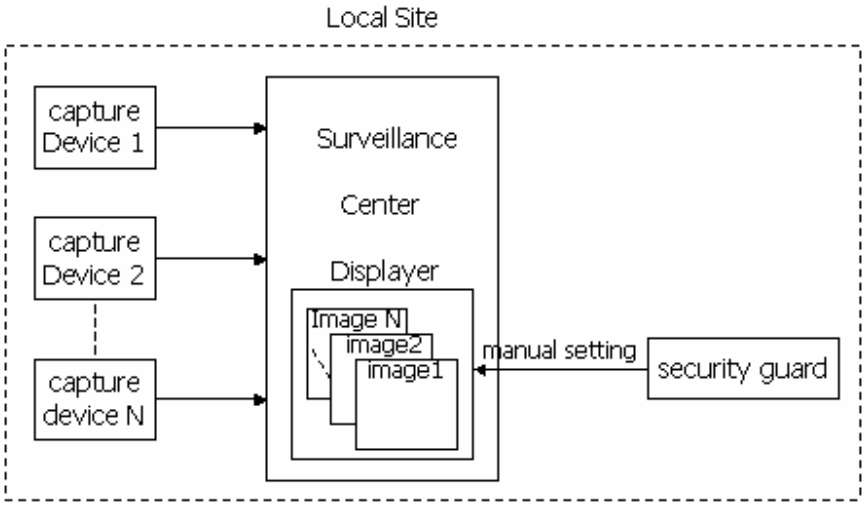


圖 3-4-3 相機換手機制(traditional)

圖 3-4-3 為傳統式的相機換手機制，displayer 可個別顯示擷取到的影像 image1~imageN，而 displayer 顯示的影像畫面需由人工設定。

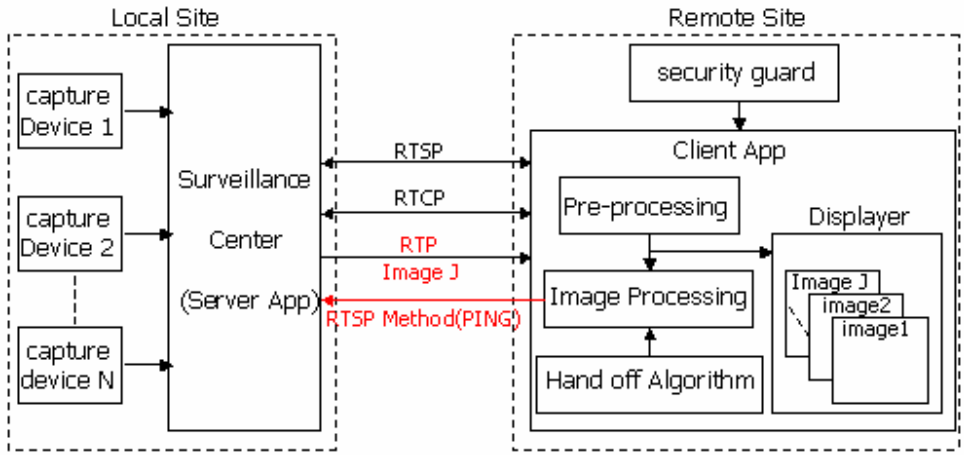


圖 3-4-4 相機換手機制(interactivel)

圖 3-4-4 為互動式的相機換手機制架構，在 Local Site 部分，監控中心透過取像設備取得影像資料。在 Remote Site 部分，安管人員透過 Client App 程式執行遠端即時檔案存取的動作。起初，Server 端會傳送某一預設的相機影像(Image J)透過串流系統傳輸資料到 Client 端。Client 端首先對接收到的資料進行串流系統相關的前置處理，經前置處理之後可得影像資料，再交給影像處理程式處理，搭配我們事先定義的相機 Hand off 機制，可決定此時要不要切換相機。如果決定要切換相機時，此時 Client 端會使用 RTSP 協定所提供的 PING Method，將所要切換的相機訊息隱藏在此 PING Message 裡，透過 Client 端的 session controller 傳送此 Method 給 Server 端。當 Server 端收到 Client 所傳送的 PING Message 之後，會去解析 PING Message 所內含的文字訊息，當它發現有切換相機的訊息之後，便會針對 Client 端所要求的相機影像，傳送此相機影像的 RTP 資料給 Client 端。如此一來，Client 端便可取得指定的相機影像資料。

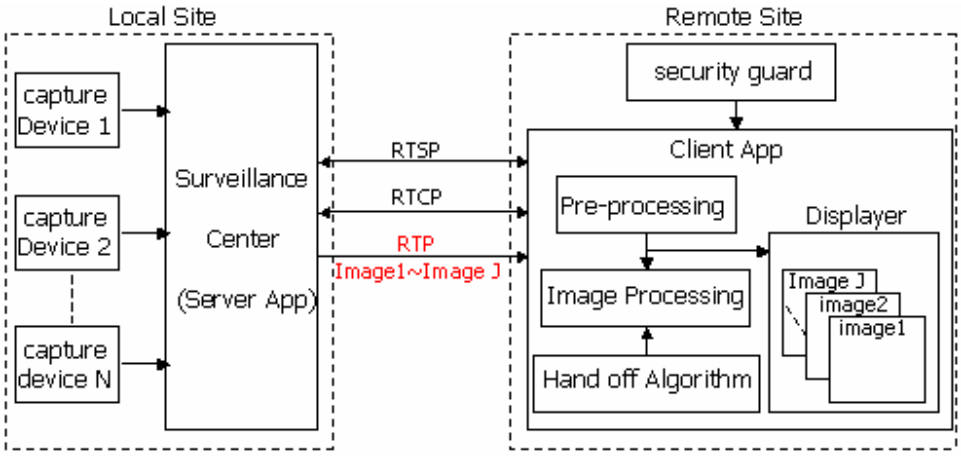


圖 3-4-5 相機換手機制(interactive2)

圖 3-4-5 為另一種互動式的相機換手機制，Server 端負責影像資料的取得，進一步的將收集到的影像資料進行 trans-coding 的動作，接著透過 Streaming Module 將資料傳輸出去。值得注意的地方為在此處所傳輸的影像訊號並非只有一個相機的影像，而是包含多個影像(Image1~ImageJ)。Client 收到 Streaming data 之後，經由前置處理取得 RGB 影像，但此時為多個影像訊號，Client 端的 Displayer 顯示的為多個相機的取像畫面。然而在 Displayer 端呈現多個影像畫面的方式會造成 Client 端的安管人員要花費很多的心力去監看各畫面的情形，因此我們改變 displayer 呈現的方式，某一時間點僅呈現一個較重要的相機取像畫面，使監控人員只需注意一個相機的影像畫面即可。而在眾多的相機影像畫面裡，如何決定 Displayer 呈現的影像。此時可將多個影像畫面送到影像處理程式處理，將處理的結果比對事前所定義的相機換手機制，如此便可以決定要呈現的相機影像。

[錄影機制]

在一般的監控系統架構下，如要進行錄影的動作時，都需要人工設定，而且所錄影的畫面在很多時間區段可能都是不重要的影像畫面。有鑑於錄影機制的互動性不足，及錄影檔中不重要的影像片段所造成的儲存空間浪費，因此我們提出一種互動式的錄影機制解決上述問題，系統架構如圖 3-4-6:

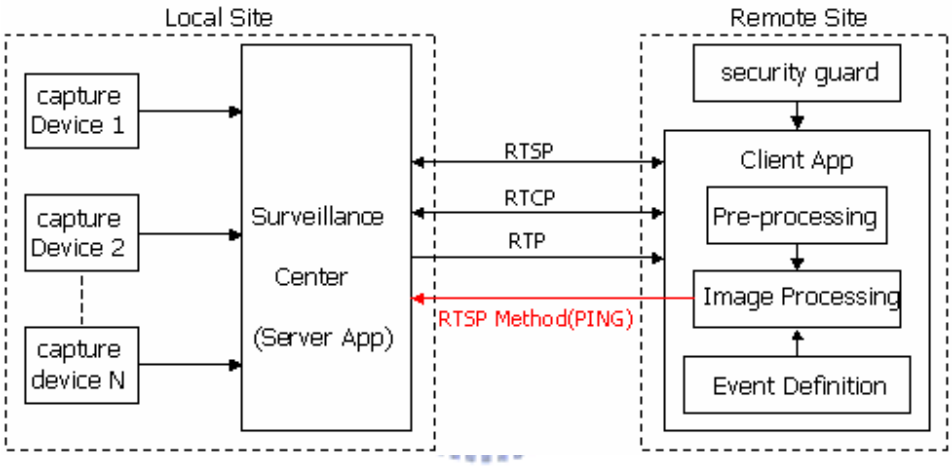


圖 3-4-6 錄影機制(1)

圖 3-4-6 為互動式錄影機制的系統架構圖，主要分為二部分，分別為 Server 端及 Client 端。Server 端負責影像資料的取得，進一步的將收集到的影像資料進行 trans-coding 的動作，接著透過 Streaming Module 將資料傳輸出去。Client 收到 Streaming data 之後，經由前置處理取得 RGB 影像，然後交由影像處理程式處理。處理之後的結果搭配事前所定義的”Event Definition”，可得知此影像有無符合事件定

義。如事件發生時，Client 端會利用 RTSP 所提供的 PING Method，將錄影訊息隱藏在此 PING Message 裡，透過 Client 端的 session controller 傳送此 Method 給 Server 端。當 Server 端收到 Client 所傳送的 PING Message 之後，會去解析 PING Message 所內含的文字訊息，當它發現有錄影訊息時，會告知 Recording Module 進行錄影的動作。然而經一段時間後，如 Client 端經影像處理發現事件已結束時，會將結束錄影訊息隱藏在 PING Message，將此 PING Message 傳給 Server 端，Server 端收到此訊息便知道事件已結束，可結束錄影的動作，如圖 3-4-7 所示：

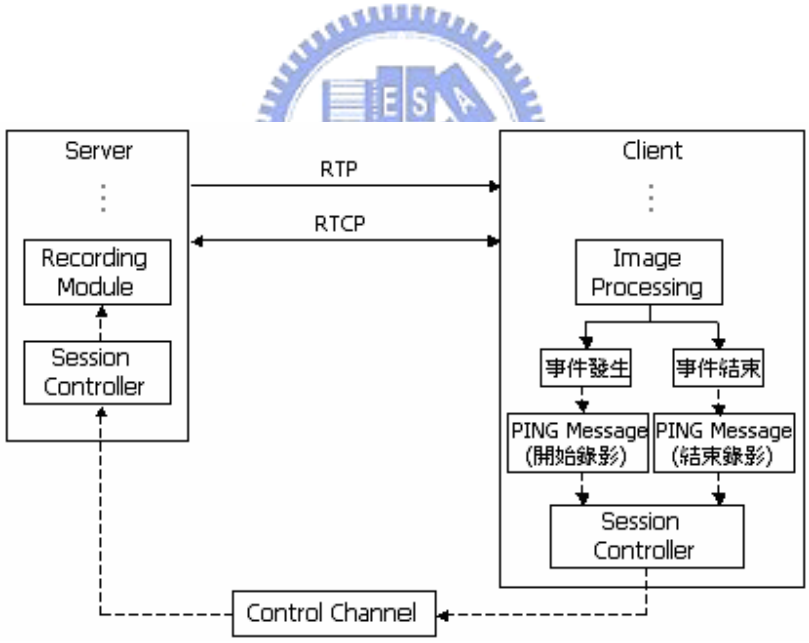


圖 3-4-7 錄影機制(2)

由上述的系統架構說明，可知此架構可達到自動錄影的功能，也就是說有事件發生才錄影。如此一來，可節省錄影檔的儲存空間及達到自動化的功能。

第四章 利用雙相機做物體平面座標定位

4.1 簡介

傳統的物件定位與追踪技術，大部分都是利用一個移動式的取像設備環境下所進行的。而在第四章中，我們將介紹一種利用二個固定式相機所發展的物件定位系統，此系統可精確的定位出移動物件在實體空間的平面座標資訊。進一步的利用此座標資訊，搭配我們所佈建的互動式監控系統，進行相機的自動換手動作，使監控平台達到互動性的要求。

4.2 物件定位

一般常見的物件定位技術大都是搭配單一影像截取裝置(例如：PTZ Camera)來進行，也就是說在同一時間點上，只對單一影像進行分析。從起初的取像，進而利用物件分割技術將移動物件(前景)抽取出來，此時所抽取出的前景影像雖大致已完成，但仍需進行影像的後置處理，如雜訊去除、漏洞填補等才可以得到較完整的前景影像。經由上述流程之後，便可以開始著手進行物件定位的動作。而到底要如何針對移動物件予以定位呢？以下簡介其基本原理，當我們完成上述定位的前置處理之後，便得到移動物件的影像，此時計算出此移動物件的中心點，找到此移動物件的中心點之後，便可依中心點的座標(x,y)，知道物件此時所

處的位置是在整張影像的何處，如此一來，便完成定位的動作。進一步的，假使要進行移動物件追蹤的話，我們可藉由連續二張或多張影像中移動物件中心點的座標資訊，進而計算出此物件運動的方向及速率，此時再搭配 PTZ Camera 的 pan、tilt 及 zoom 的功能，移動 Camera 鏡頭的角度，使最新估測之移動物件中心點呈像在影像的中心位置。上列文字敘述是針對移動物件的定位及追蹤說明，以下列出此系統流程圖[5]：

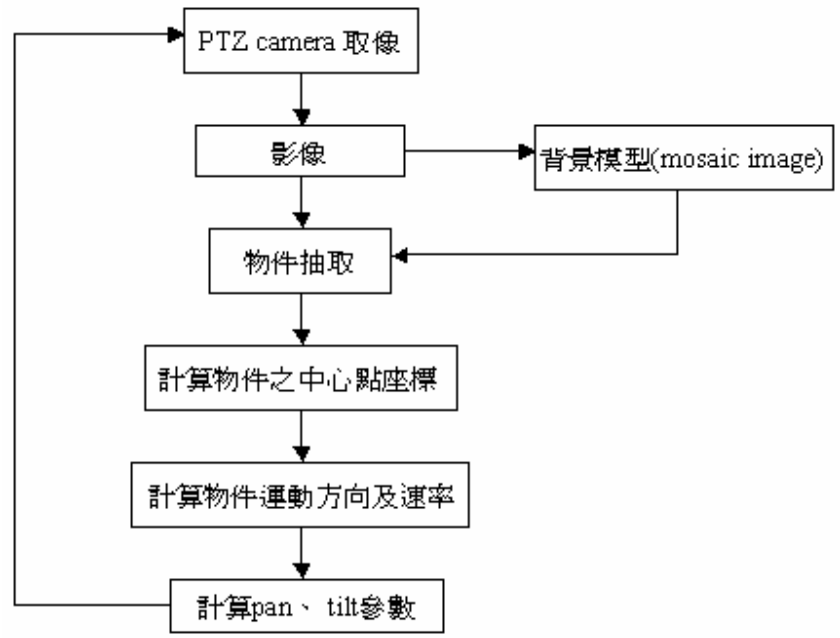


圖 4-2-1 物件定位與追蹤流程圖

圖 4-2-1 為物件定位與追踪流程圖，主要是針對移動式取像設備所發展出的。但是一般固定式的取像設備並不具備可移動的特性，卻又要如何進行物件的定位及追蹤。下面我們將探討在單一相機及雙相機情形下，物件的定位及追蹤。

[單一固定式取像設備情形]

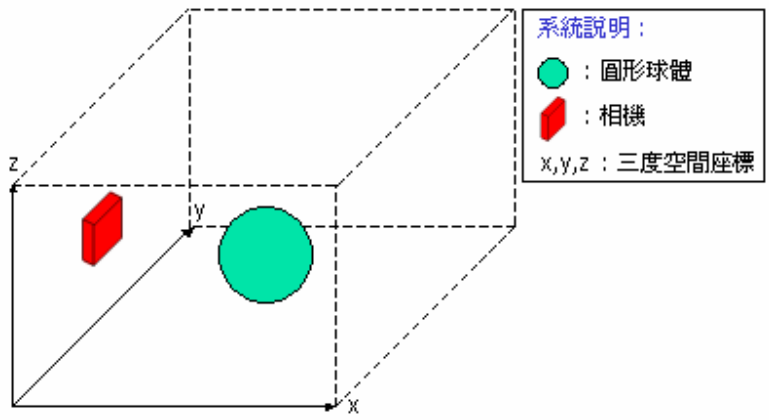


圖 4-2-2 單一固定式取像系統(立體圖)

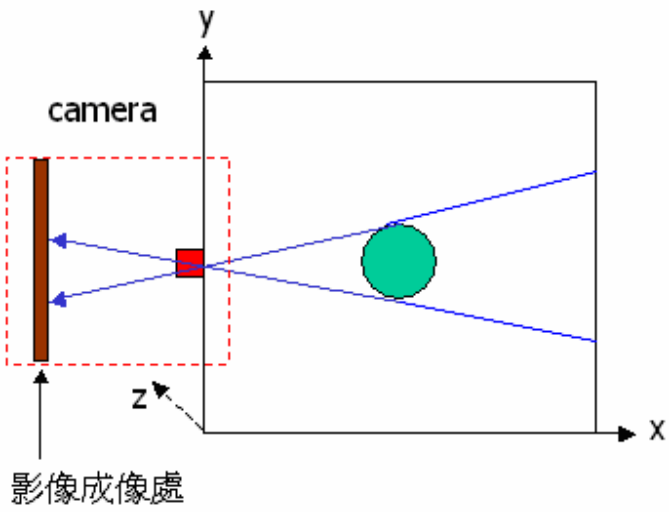


圖 4-2-3 單一固定式取像系統(俯視圖)

圖 4-2-2 為單一固定式取像系統的立體圖、圖 4-2-3 為單一固定式取像系統的俯視圖，x 軸、y 軸代表著三度空間的平面分量資訊，z 軸代表三度空間的垂直分量，空間內的物件為圓形球體，相機擺放位置如圖 4-2-2 所示，平行於 z、y 平面。如圖 4-2-3 所示，空間內的圓形球體藉

由 camera 的取像及成像過程之後，此時 camera 便有此物件資訊的影像 。Camera 的輸出影像如圖 4-2-4：

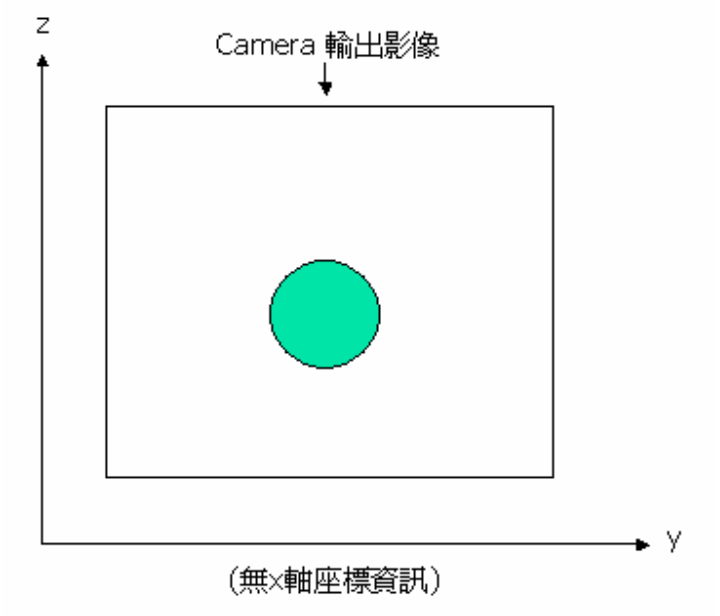
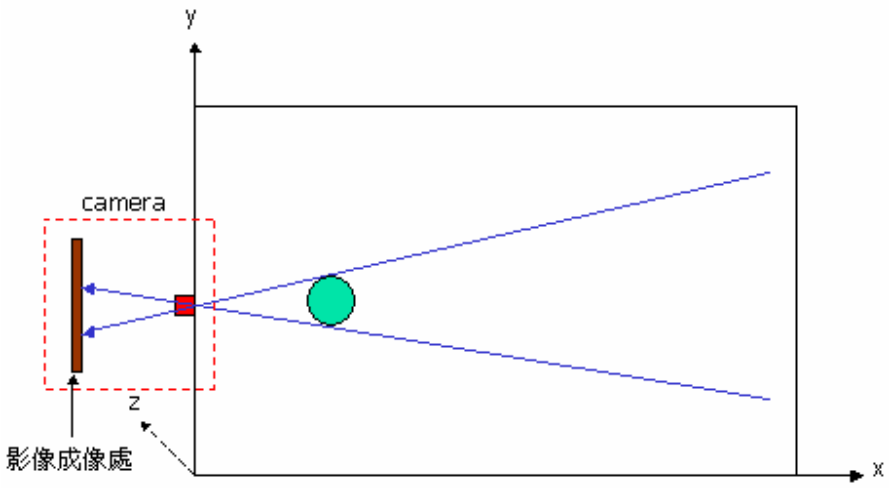


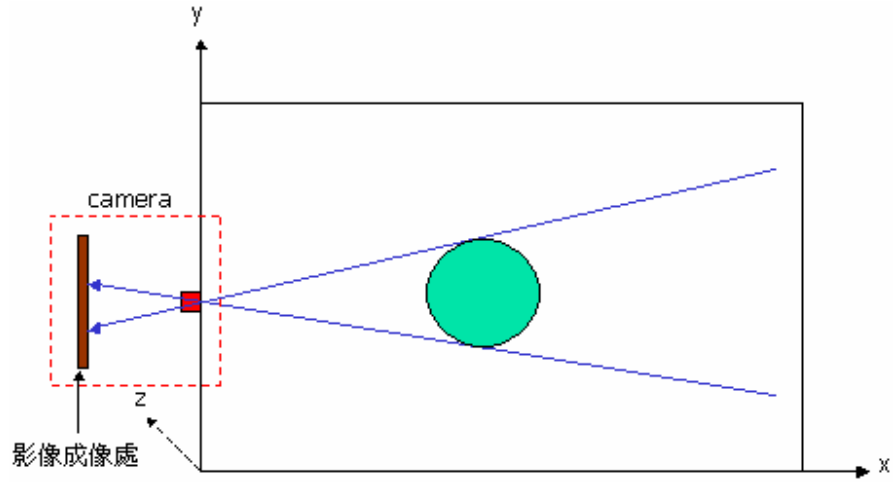
圖 4-2-4 camera 輸出影像

圖 4-2-4 為 camera 的輸出影像，由此圖可清楚明白的看到單一影像只能呈現出物件所處位置的 y 軸及 z 軸資訊，但是卻沒有平面分量資訊中 x 軸的分量。很不幸地，缺少這個 x 軸的資訊的話，我們並沒有辦法進行後續的平面座標定位，只可以針對此物件的 y 軸分量資訊予以定位。進一步的對此問題進行探討，一個實體物件在三度空間理應具有 x 軸、y 軸、z 軸三個分量，但藉由 camera 的取像及成像機制之後，x 軸分量資訊遺失了。也就是說在某些條件下，實體物件在三度空間的 x 軸分量為 1 單位或 2 單位等等，對 camera 而言，取像後的輸出影像是一樣的，這樣的情形會使得定位問題無法執行，以圖 4-2-5 說明此問題。

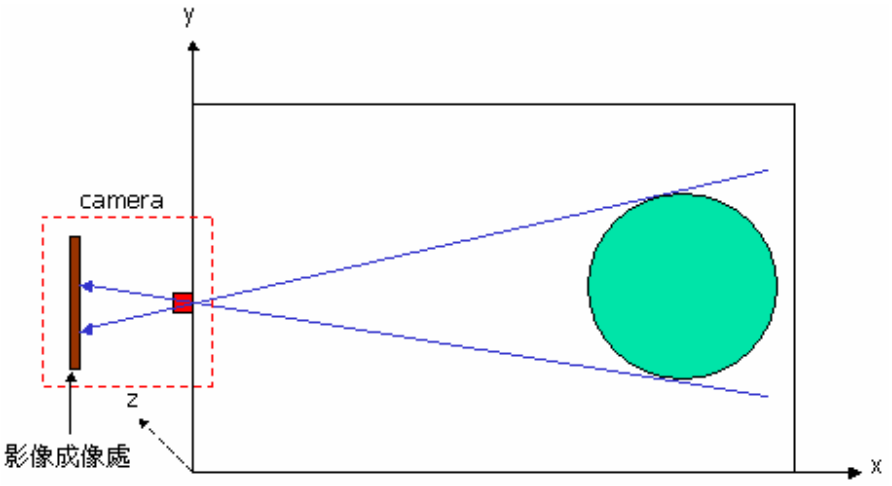
［物件深度問題］



(a)



(b)



(c)

圖 4-2-5 物件深度問題

藉由圖 4-2-5，說明在使用單一取像設備的情形下，物件深度的問題。上圖(a)、(b)、(c)三種情形的圓形球體雖然分別座落於三個不同的三度空間位置，但其實對取像裝置而言，三者並沒有任何不同，三個輸出影像也都一樣。會造成這樣的原因是因為取像裝置在取像時，三度空間的物體會投影到二度空間，遺失了 x 軸維度的資訊，造成上圖之三個圓形球體雖然其位置不一樣(x 軸分量不同)，但 camera 輸出影像卻相同，這個現象就是所謂的”物件深度問題”。

即然利用單一取像裝置進行物件定位時，會發生物件深度問題，以致於沒有足夠的資訊對移動物件定位，因此我們利用二個或多個取像裝置來進行物件定位的動作，此方法不僅順利的解決物件深度問題，也可以著手進行物件定位的動作，以下將逐步說明此系統架構是如何的解決上述相關問題。

[雙相機系統]

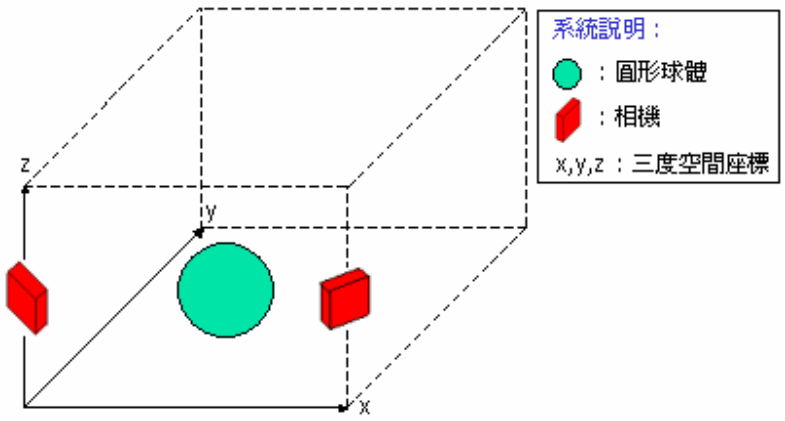


圖 4-2-6 雙相機系統(立體圖)

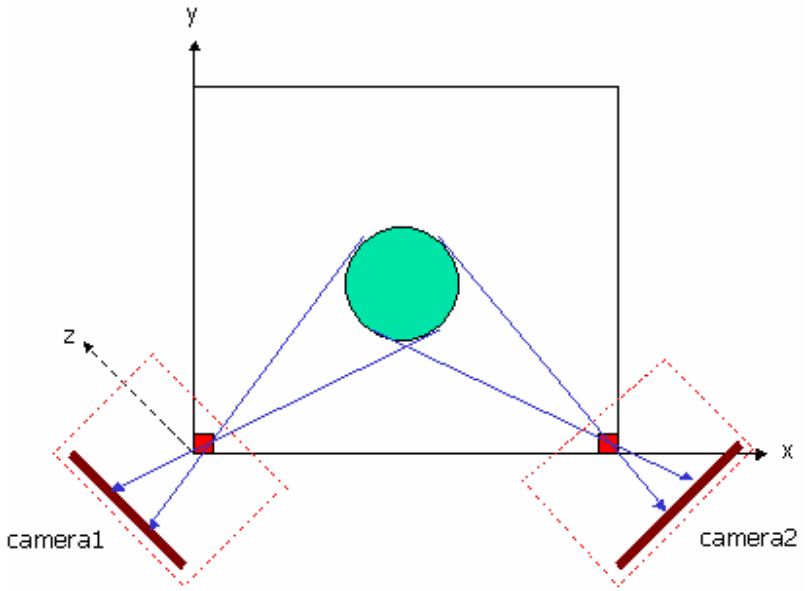


圖 4-2-7 雙相機系統(俯視圖)

圖 4-2-6、圖 4-2-7 分別為雙相機系統的立體圖及俯視圖，此系統佈建情形和單一固定式取像設備取像情形差不多，唯一的差別是多了一個取像設備。以下將說明在此系統下，要如何達到移動物件的定位，及在進行定位時可能遭遇的相關問題。

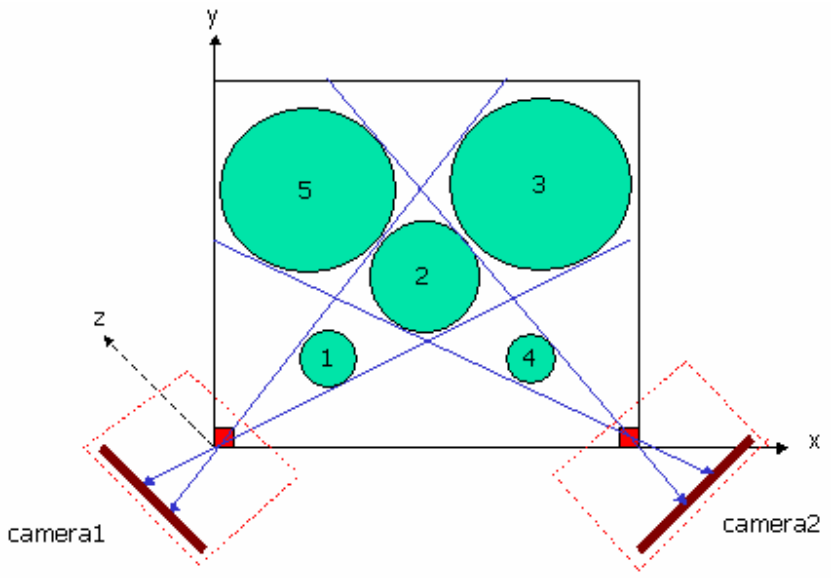


圖 4-2-8 雙相機系統深度問題

首先我們先探討物件深度的問題，物件的分佈情形如圖 4-2-8 所示。針對 camera1 而言，圓形球體 1、圓形球體 2、圓形球體 3 三者對 camera1 來說，經取像及成像過程後，三者的輸出影像都是相同的，也就是說針對此方向的物件分佈，是無法分辨其大小及深度的，也就是有”物件深度問題”。同樣地，針對 camera 2 而言，圓形球體 2、圓形球體 4、圓形球體 5 三者對 camera2 來說，三者的輸出影像也都是相同的，面對此方向的物件分佈，仍無法分辨其大小及深度的，仍然有”物件深度問題”。雖然由前面的說明知道二個 camera 都有物件深度的問題，也就是說二者都無法獨自的進行物件定位的動作，但是如果同時利用二 camera 進行物件定位動作的話，卻是可行的。以圖 4-2-8 的物件分佈為例，camera1 雖不能分辨圓形球體 1、2、3 之間的深度及大小，

camera2 也不能分辨圓形球體 2、4、5 之間的深度及大小，但用這二 camera 可同時取像到圓形球體 2，於是我們便可藉由 camera1 掌握的圓形球體 2 的二維資訊，加上 camera2 握有的圓形球體 2 的二維資訊，由這二 camera 提供的二維資訊去找尋球體 2 的平面座標資訊。下面將更詳盡的說明此系統是如何進行物件定位的工作。

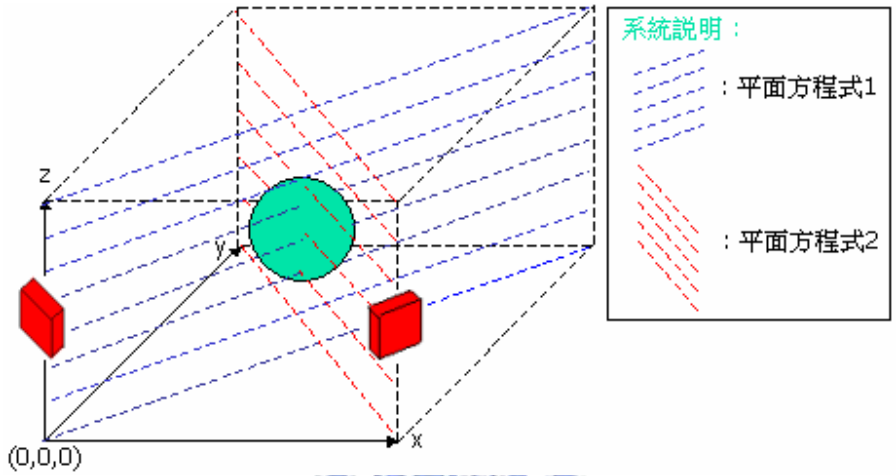


圖 4-2-9 物件定位(立體圖)

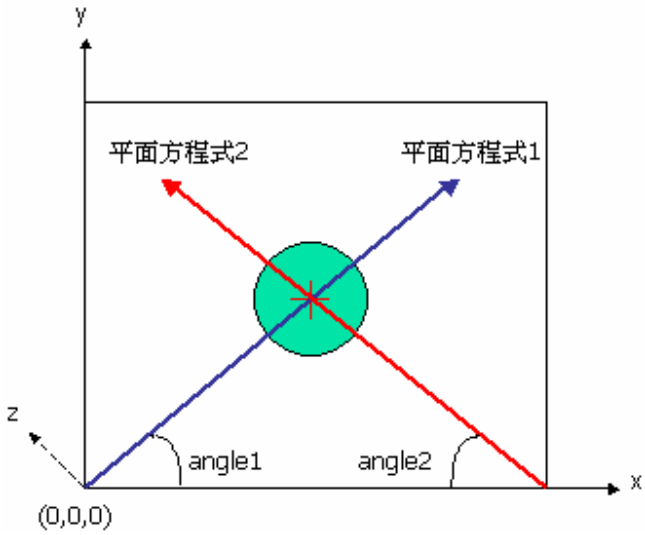


圖 4-2-10 物件定位(俯視圖)

以下提出在雙相機系統情形下，移動物件的平面座標定位技術，並以圖 4-2-9、圖 4-2-10 來說明其基本原理。首先，camera1 截取它所處位置的影像，進一步的將影像中的移動物件抽取出來，並計算此物件的中心點，再搭配 camera1 在此空間擺放的位置資訊，可算出在此空間中 angle1 的角度，接著再利用這個角度的資訊，建立一由參考點(0,0,0)通過物件中心的平面方程式 1。camera2 執行的運算和 camera1 一樣，由影像中抽取出移動物件，計算物件中心點，配合 camera2 在此空間擺放的位置資訊，計算出 angle2 的角度，建立一通過物件中心的平面方程式 2。經由上述流程之後，分別由二個 camera 資訊建立了二平面方程式，有了平面方程式之後，便可利用克拉碼法則(Cramer Rule)，解出二平面方程式的解，得到的(x,y)資訊即是物件在此空間的平面座標(忽略 z 軸座標)，如此一來便完成物件定位的工作。上述的文字說明是在二個取像裝置下，物件的定位概念及基本原理，接著我們將列出實際的數學運算步驟與流程。

[angle1、angle2 的求法]

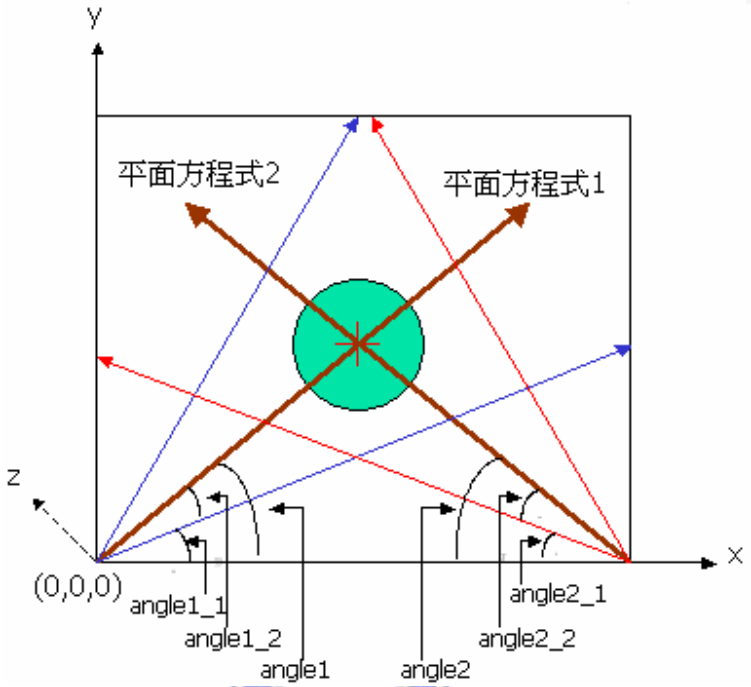


圖 4-2-11 角度計算(1)

我們以圖 4-2-11 說明如何求得 angle1、angle2 及建立二平面方程式。圖中的二條藍色射線的夾角區域表示左下 camera 鏡頭所能截取到影像的區域，二條紅色射線的夾角區域表示右下角 camera 鏡頭所能截取到影像的區域，夾角區域的大小取決於 camera 的廣角角度。由上圖知道 angle1 是由 angle1_1 和 angle1_2 所組成，angle2 是由 angle2_1 和 angle2_2 所組成。angle1_1 代表著左下角 camera 鏡頭的夾角區域邊界和實體空間 x 軸之間的夾角，此角度取決於相機的擺放角度，angle2_1 代表著右下角 camera 鏡頭的夾角區域邊界和實體空間 x 軸之間的夾角，此角度取決於相機的擺放角度。由上說明可知相關角度求法：

$$angle1 = angle1\_1 + angle1\_2 \quad (4.1)$$

$$angle2 = angle2\_1 + angle2\_2 \quad (4.2)$$

angle1_1、angle2_1 取決於相機的擺放角度，angle1_2、angle2_2 則是由移動物件中心點位置決定。以下面例子說明 angle1_2、angle2_2 的求法：

[angle1_2、angle2_2 求法]

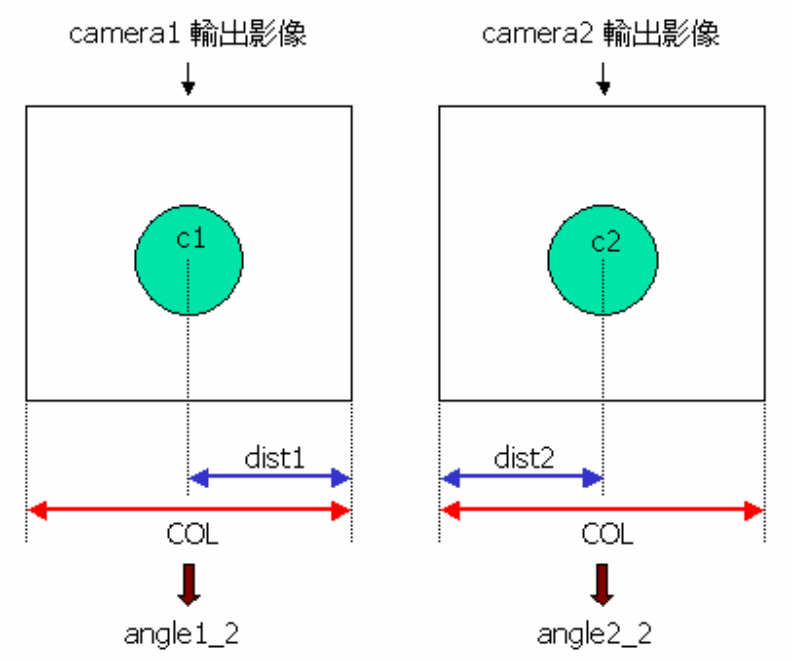


圖 4-2-12 角度計算(2)

如圖 4-2-12 所示，先對 camera1、camera2 輸出影像進行移動物件抽取的動作，取得移動物件的影像之後，計算物件的中心點 c1、c2，根據物件中心點座標可算出中心點 c1、c2 到影像左右邊界的水平分量距

離 dist1、dist2，然後再依據 dist1、dist2 和影像寬度(COL)之比例，搭配相機廣角角度，可求得 angle1_2、angle2_2 的角度，配合上述所說的 angle1_1、angle2_1 的值便可得知 angle1、angle2。

當順利的求得 angle1、angle2 的角度之後，便可以很容易的建立二通過物件中心的平面方程式，接著再將二平面方程式利用克拉碼法則(Cramer Rule)求得二方程式之解 x,y，此時的(x,y)(忽略 z 軸座標)即是移動物件在此三度空間的平面座標，經上述運算程序之後，便完成物件定位的動作。完整的物件定位流程如圖 4-2-13 所示：

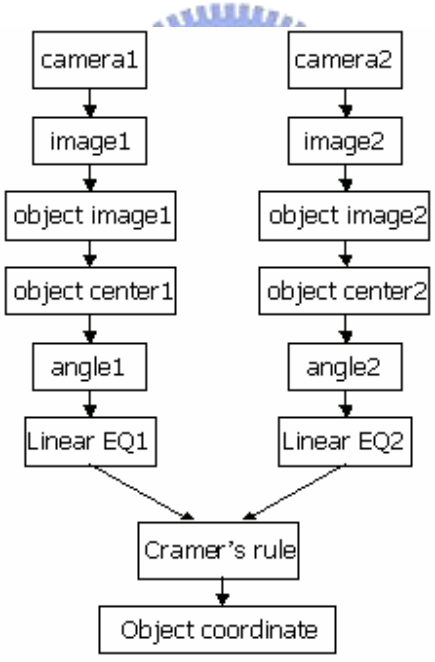


圖 4-2-13 物件定位流程圖

[可定位區間]

經由上述說明，我們知道如何針對一移動物體進行平面定位的動作，然而當中有一點值得注意，angle1_1、angle2_1 代表著二相機的擺放位置資訊，因此不同的擺放位置影響了 angle1_1、angle2_1 的角度，也連帶影響平面定位系統的可定位區，如圖 4-2-14 所示：

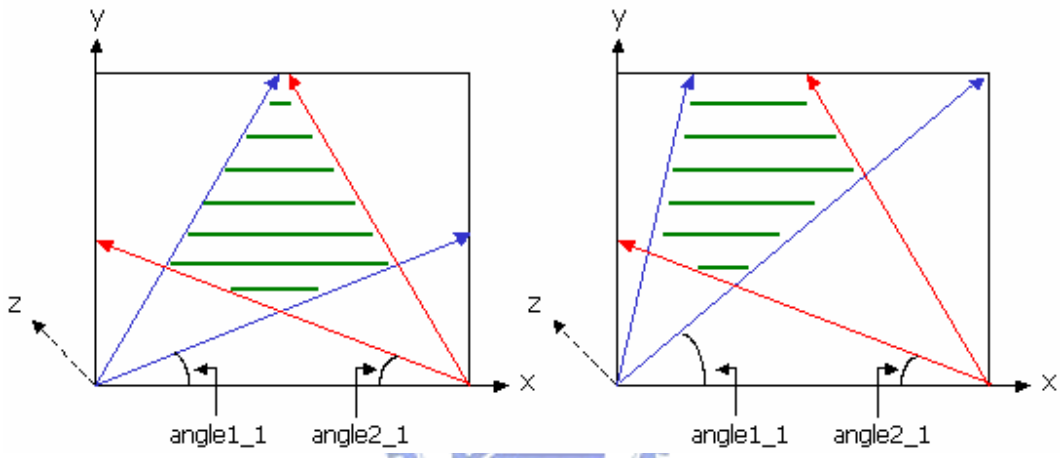


圖 4-2-14 可定位區間

如圖 4-2-14 所示，左圖為我們原始設計的相機擺放位置，二相機鏡頭分別正對於二對角。右圖為將左下角相機的鏡頭以逆時針方向偏移某一角度的情形。圖中綠色線條區域為此定位系統的〝可定位區〞，如圖所示，針對不同的相機擺放位置，造成了不同的可定位區。

[平面定位系統參數說明]

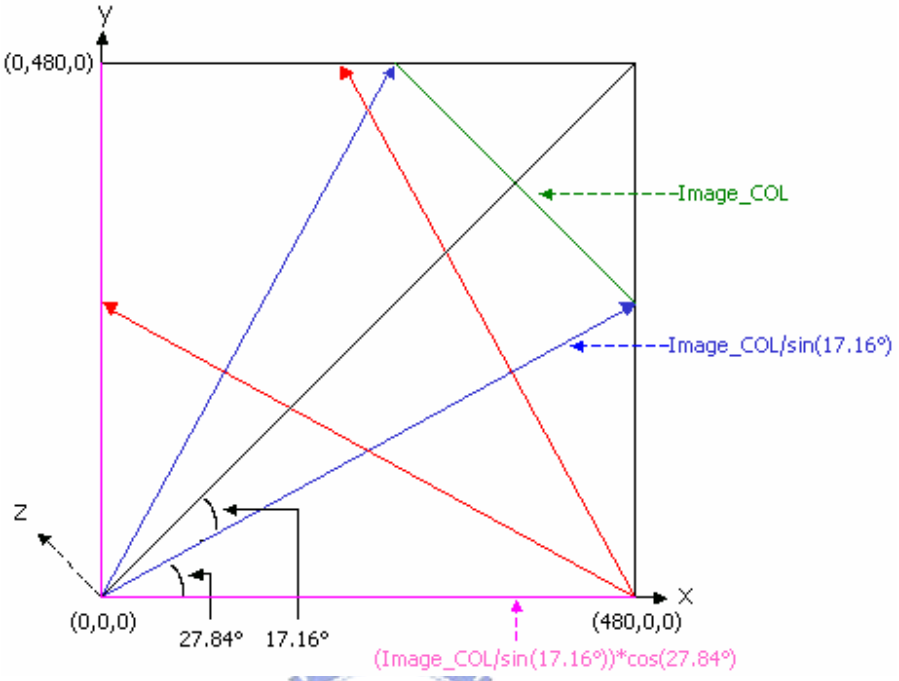


圖 4-2-15 平面定位系統參數說明

藉由圖 4-2-15 來說明進行平面定位時重要的參數。目前系統所使用的相機廣角角度為 34.32 度、輸出影像 size 為 320x240，定位空間的平面為正方形平面，二相機的擺放位置分別座落於正方形的二角落且將相機鏡頭正對於各自的正方形對角。由上資訊可得知前述定義之 angle1_1、angle2_1 角度為 27.84 度。另外，系統的定位座標以平面座標(0,0)~平面座標(480,480)來表示，此值是由相機的輸出影像寬度(Image_COL)，及 angle1_1、angle2_1 及相機廣角角度的值利用簡單的三角函數運算所求得的。上述參數整理於表 4-2-1：

表 4-2-1 平面定位系統參數

相機廣角角度	34.32 度
相機輸出影像之寬(Image_COL)	320
相機輸出影像之高(Image_ROW)	240
定位座標	(0,0)~(480,480)

[平面定位系統運算流程]

上述文字為雙相機平面定位系統的說明，以下將以步驟的方式說明實際的運算流程。

Step1：

利用背景相減法，算出二取像相機的前景影像 foreground_1、foreground_2。

Step2：

計算二前景影像的物件中心點 c1、c2，運算如下：

$$
\begin{aligned}
&for\ (foreground(i,j)!=0)\\
&\{\\
&\quad temp\_i = temp\_i + i;\\
&\quad temp\_j = temp\_j + j;\\
&\quad num\_pixel = num\_pixel + 1;\\
&\}\\
&center\_i = \frac{temp\_i}{num\_pixel};\\
&center\_j = \frac{temp\_j}{num\_pixel};
\end{aligned}
$$

center_i 表示此物件中心在影像中的水平分量座標，center_j 表示此物件中心在影像中的垂直分量座標。

Step3：

計算二物件中心點到影像水平邊界的距離 dist1、dist2，進而求得 angle1_2、angle2_2，運算如下：

$$
\begin{aligned}
&\text{c1的座標爲(c1\_i,c1\_j)}\\
&\text{c2的座標爲(c2\_i,c2\_j)}\\
&\text{dist1=COL-c1\_i}\\
&\text{dist2=c2\_i}\\
&\text{angle1\_2=angle\_function}\left(\frac{\text{dist1}}{\text{COL}}\right)\\
&\text{angle2\_2=angle\_function}\left(\frac{\text{dist2}}{\text{COL}}\right)
\end{aligned}
$$

其中 angle1_2、angle2_2 是由 angle_function 所求得，以下舉例說明 angle_function 的運算。如相機廣角角度為 34.32 度，則：

$34.32^\circ = angle\_function\left(\frac{320}{320}\right)$ 、$17.16^\circ = angle\_function\left(\frac{160}{320}\right)$ 、

$0^\circ = angle\_function\left(\frac{0}{320}\right)$ 。

Step4：

利用給定的 angle1_1、angle2_1 的角度資訊及 step3 所算出的 angle1_2、angle2_2 資訊，求得 angle1、angle2。利用此二角度建立二條通過物件中心座標的直線方程式(註：原始概念為建立二平面方程式，但實際運算時只需建立二直線方程式即可)，建立方式如下：

直線方程式1： $\frac{y-0}{x-0} = \tan(angle1)$	$=> \tan(angle1)*x - y = 0$
直線方程式2： $\frac{y-0}{x-480} = -\tan(angle2)$	$=> \tan(angle2)*x + y = \tan(angle2)*480$

Step5：

利用 Cramer Rule 解二直線方程式的交點，此交點即為物件在此空間的平面座標。

4.3 相機之換手機制

在雙相機系統架構下，可輕易對移動物件進行定位的動作，然而物件定位所帶給我們的好處並不僅僅於此，我們可進一步的利用物件在某一空間的位置資訊，進行相機的換手(Hand Off)機制，使系統能針對物件的移動，找出最佳視野(view)的相機，使後續影像處理所需的輸入影像更符合其需求。以下介紹二種相機換手機制。

[物件清晰度為基準]

當物件在某一空間移動時，礙於相機的廣角角度因素，相機鏡頭針對某些特定的位置，取像時可能取不到。或是在某些情形下，camera1 針對物件取像時，對物件的呈現可能大小、位置皆適中；camear2 取像時，對物件的呈現可能太大或太小，或者使物件座落於影像中的角落位置，此時我們當然希望以 camera1 的影像為主，以利進行後續影像處理。以下舉出幾個相機換手機制的情形：

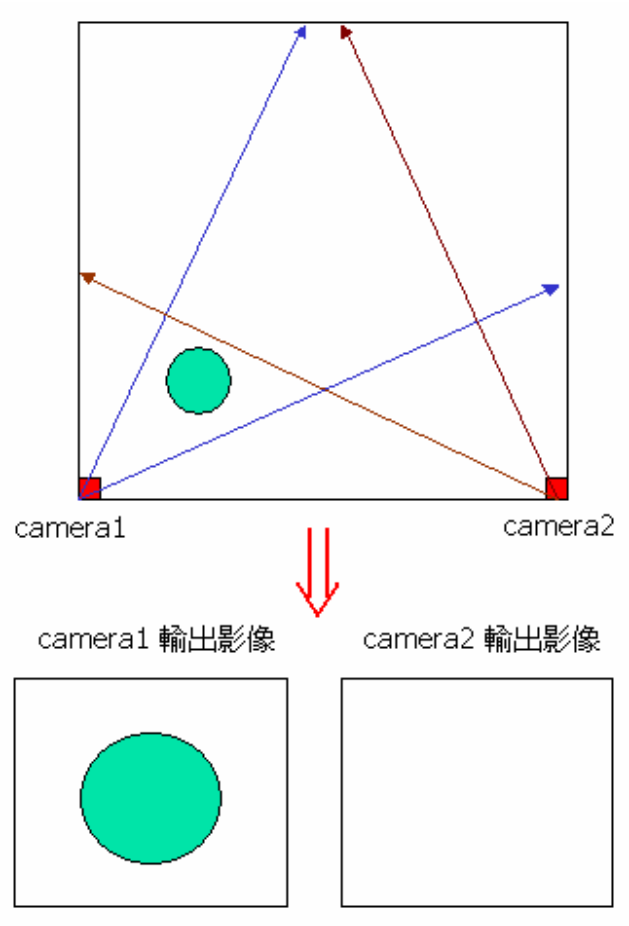


圖 4-3-1 相機換手機制(scenario1)

假設物件的分佈情形如圖 4-3-1 所示，此時 camera1 所擷取的影像，對物件的呈現是大小、位置皆適中的。camera2 卻因為礙於相機的廣角角度因素使其無法補抓到此物件，使其輸出影像完全沒有物件的資訊，此時 camera1 的位置將是較好的取像點。

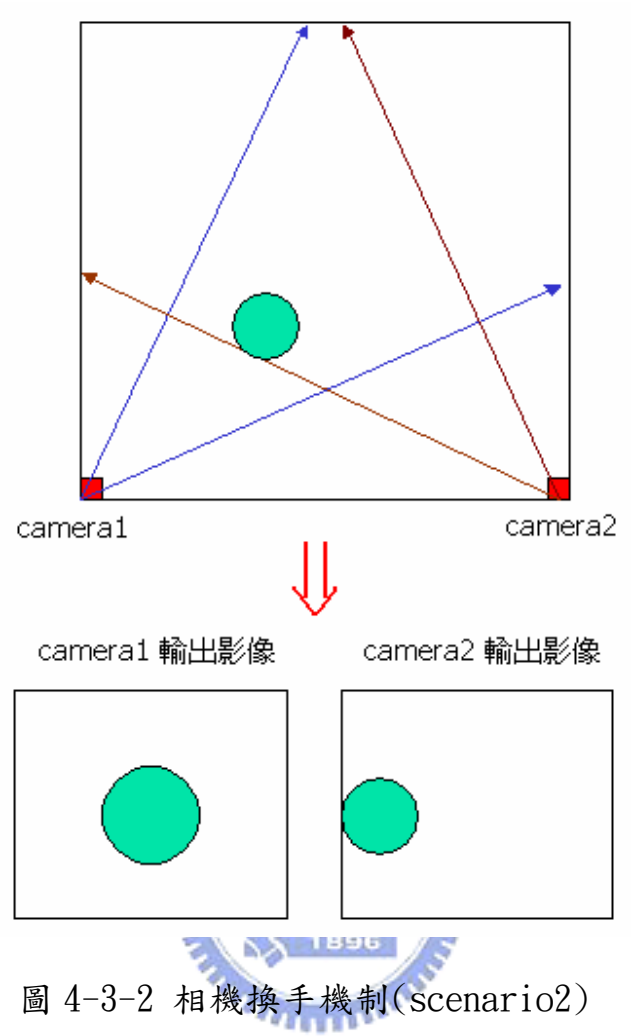


圖 4-3-2 相機換手機制(scenario2)

我們考慮另一種情況，物件所處位置如圖 4-3-2 所示，camera1 對此位置的物件取像很適合，輸出影像中物件的大小、位置皆很好；反觀 camera2 對此位置的物件呈現，其取像位置、大小並不是很恰當，使物件座落於影像中的邊界處、且物件大小較小，此時 camera1 的位置將是較好的取像點。以下列出上述相機換手機制：

Hand off algorithm:

$$for\ (object\_image1(i,j)!=0)\{num\_pixel1=num\_pixel1+1;\}$$

$$for\ (object\_image2(i,j)!=0)\{num\_pixel2=num\_pixel2+1;\}$$

$$if\quad (num\_pixel1>num\_pixel2+threshold)$$
$$hand\ off\ =>camera1;$$

$$else\ if\quad (num\_pixel2>num\_pixel1+threshold)$$
$$hand\ off\ =>camera2;$$

$$else\quad hold\ \ on\ \ previous\ \ state;$$

object_image1、object_image2 分別代表由 camera1、camera2 原始影像中，抽取出的移動物件影像(前景)，接著我們說明上述演算法的運算流程。首先計算 object_image1、object_image2 中，非零的像素總數(即物件呈像的大小)，此值紀錄於 num_pixel1、num_pixel2。接著比對二 camera 取出的物件大小，如二物件大小差距大於給定的 threshold 值的話，此時可決定主要的取像 camera，如二物件大小差距小於 threshold 的話，表示二 camera 取出的物件大小差不多，可不必執行換手的動作，保持前一個狀態的主要取像 camera 即可。

[物件座標為基準]

除了以物件清晰度為基準所進行的相機換手機制外，更可依物件的座標資訊進行相機換手的動作。以下面的例子說明：

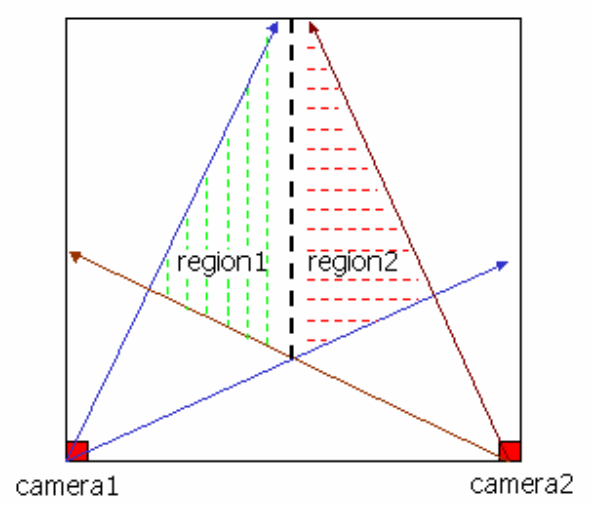


圖 4-3-3 相機換手機制(scenario3)

我們以二台取像設備為例，如圖 4-3-3 所示，region1 加上 region2 的地方表示此系統架構下可定位的區域，此可定位區域可隨意定義出子區域以提供相機換手的資訊。如上圖所示，將定位區域分割成 region1 及 region2，定義 region1 的主要取像相機為 camera1，region2 的主要取像相機為 camera2。如有一物件經定位後其中心點座標位於 region1 時，此時將主要取像相機設為 camera1；反之如位於 region2 時，便將主要取像相像設為 camera2。以下列出上述相機換手機制：

Hand off algorithm :

given region 1,region 2,...regionN information

given correspondence between region and camera

if (*object _center located at* region 1) {*hand off => correspondent camera*;}

if (*object _center located at* region 2) {*hand off => correspondent camera*;}

⋮

if (*object _center located at* region N) {*hand off => correspondent camera*;}

針對此相機換手機制的實現，需要先將此系統架構下的可定位區域切割成 N 個子區域，接著定義區域和主要取像相機的對應關係，然後執行物件定位的動作，如物件中心點座標位於某一 region 內，便將主要取像相機設為對應的相機。

第五章 手勢辨識

5.1 簡介

在影像處理的範疇裡，物件辨識一直是人們研究的熱門課題，而當中又以手勢辨識為最多人所研究。從早期利用戴手套的方式，藉由手套上光點的分佈，進行光點比對的手勢辨識。後來有人利用手部的幾何分佈、色彩特徵等資訊進行手勢辨識的運算。最後有人利用隱藏式馬可夫模式、Modified Fourier Descriptors 方式來進行手勢的辨識[6]。上述所提的都是過去及近年手勢辨識的運算方式，在此我們提出一種利用手勢特徵點的幾何分佈、搭配手部色彩資訊所進行的手勢辨識運算。進而將此手勢辨識系統實現在我們所佈建的即時互動式監控平台上，使其提供第三章所提的事件定義，使平台達到互動性的要求。

5.2 特徵點擷取

當我們在進行物件辨識的工作時，總希望物件本身有其特殊的地方，即所謂的”特徵點”，顧名思義為有別於一般的點。一般的特徵點為具有邊(edge)、角(corner)、色彩(color)等性質，有這些性質的點即為特徵點。然而要如何取得影像中的特徵點，一般常見的演算法有Harris Corner Extraction、Haar Wavelet、Garbor Wavelet、Daubechies

Wavelet 等等[7]。而因為在後續手勢辨識的應用是利用 corner 的資訊進行辨識的工作，所以在此則強調 corner point 的擷取，以下將介紹 Harris Corner Extraction 演算法[8]、[9]：

[Harris corner extraction algorithm]

以下將以步驟流程的方式說明此演算法是如何的進行corner point擷取的動作。下圖為 Harris corner extraction algorithm 運算流程圖：

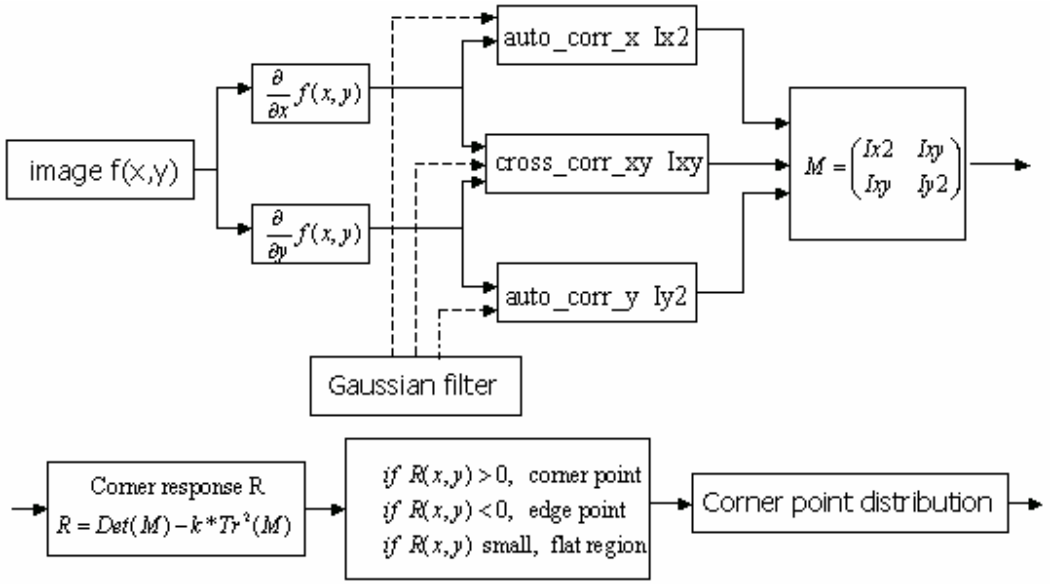


圖 5-2-1 Harris corner extraction 運算流程圖

圖 5-2-1 為 corner point 擷取的運算流程圖，下面以 step1~step6 來說明整個運算流程。

step1：對原始影像 $f(x,y)$ 進行 x 軸方向的偏微分，運算後的影像定義為

$Ix(x,y)$ 。[$Ix(x,y)=\frac{\partial f(x,y)}{\partial x}$]

step2：對原始影像 $f(x,y)$ 進行 y 軸方向的偏微分，運算後的影像定義為

$Iy(x,y)$ 。[$Iy(x,y)=\frac{\partial f(x,y)}{\partial y}$]

step3:給定一Gaussian filter的係數g,搭配此係數求取 $Ix(x,y)$、$Iy(x,y)$ 的 auto-correlation 影像 $Ix2$ 、 $Iy2$ 及 cross-correlation 影像 Ixy 。[Ix2=conv2(Ix.^2,g)]、[Iy2=conv2(Iy.^2,g)]、[Ixy=conv2(Ix.*Iy,g)]

step4：利用 $Ix2$ 、 $Iy2$ 、 Ixy 的資訊，建立矩陣 M。[$M=\begin{pmatrix} Ix2 & Ixy \\ Ixy & Iy2 \end{pmatrix}$]

step5：利用矩陣 M 的資訊，建立 Corner response 矩陣 R。

[$R=Det(M)-k*Tr^2(M)$]

step6：利用 Corner response 矩陣 R 的資訊，決定影像中 corner point 的位置。

如果 R(x,y)>0，則表示影像中(x,y)的地方為一 corner point。

如果 R(x,y)<0，則表示影像中(x,y)的地方為一 edge point。

如果 R(x,y)值很小，表示影像中(x,y)的地方為一 flat region。

由上述 step1~step6 的運算流程，可針對某一影像，取出影像中的 corner point。以下顯示一實際的例子：

[Harris Corner Extraction 例子]

圖 5-2-2 原始影像

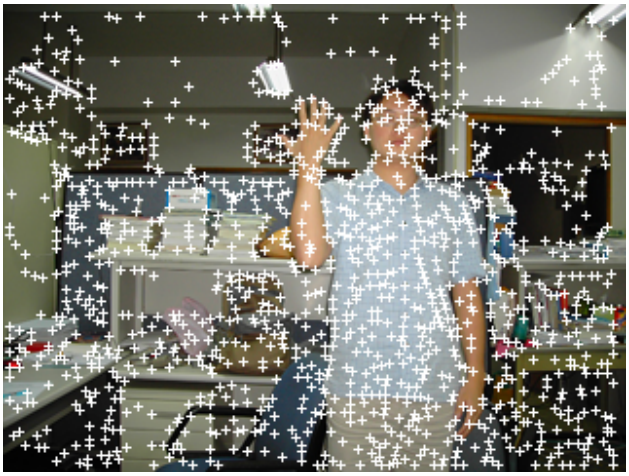


圖 5-2-3 corner point distribution 影像

圖 5-2-2 為原始輸入影像，圖 5-2-3 為原始影像經 Harris Corner Extraction 演算法所求得的 Corner point 分佈。當中有一點是需要注意的，雖然由 Harris corner extraction 所定義的 corner point 條件為 corner response $R(x,y)>0$，但此時找出的 corner point 可能太多(圖

5-2-3)。因為計算出的 corner response $R(x,y)$ 針對不同的景物，所反應的 corner 強度是不同。如方形的桌角點其 corner 強度是很大的，而較圓滑形的桌角點其 corner 強度可能較小。因此在實際進行 corner point 的擷取時，可指定一 threshold，定義 R(x,y)>threshold 時才為 corner，如此可消除 corner 強度小的點(圖 5-2-4)。

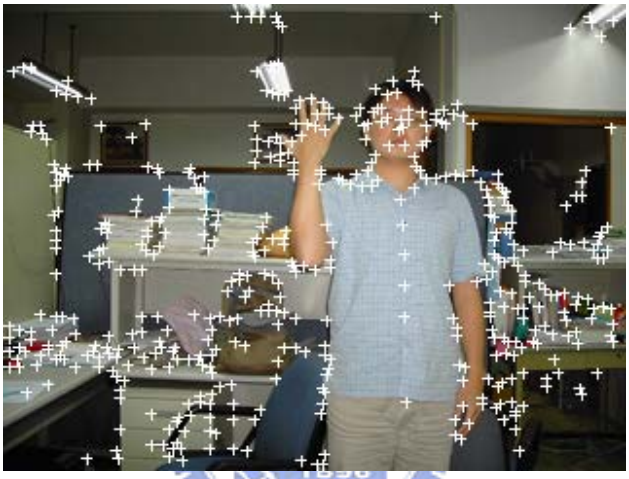


圖 5-2-4 corner point(thresholding)

5.3 樣板比對

樣板比對是一種將擷取到的影像資料與資料庫內的既有樣板資訊作比對的動作，找出影像中相似於樣板的地方。然而資料庫內的原始樣板可能未必是比對時的最佳選擇，此時可加入某些參數如 scale、rotation，控制樣板的放大縮小、旋轉，使其更符合影像中欲比對的物件格式。然而在樣板比對時，怎樣的比對結果是所謂的”最相似”，一

般常見的metric為計算影像子區塊和樣板之間的difference或cross-correlation的值,以定義二者之間的相似程度。以下列出這二個metric的數學運算式：

$$D(x,y)=\sum_{i=1}^{tmp\_col}\sum_{j=1}^{tmp\_row} abs\left(templet(i,j)-f(i+x-\frac{tmp\_col}{2},j+y-\frac{tmp\_row}{2})\right) \quad (5.1)$$

$$C(x,y)=\sum_{i=1}^{tmp\_col}\sum_{j=1}^{tmp\_row}\frac{[templet(i,j)-\mu_{templet}]*[f(i+x-\frac{tmp\_col}{2},j+y-\frac{tmp\_row}{2})-\mu_{f(x,y)}]}{tmp\_col*tmp\_row} \quad (5.2)$$

方程式(5.1)、(5.2)中的$D(x,y)$、$C(x,y)$分別代表著樣板中心移到影像位置(x,y)的地方時，影像子區塊和樣板影像的像素difference及cross-correlation。以difference來看，D(x,y)的值越小，表示樣板和在位置(x,y)的影像子區塊相似度越高。以cross-correlation來說，C(x,y)值越大,表示樣板和在位置(x,y)的影像子區塊二者相似度越高。

比對方式：

樣板比對的目的是找尋影像中相似於樣板的地方，然而如何的比對、找尋策略也很重要，因為Search Strategy關係著計算量、計算時間及準確度的問題,應依使用環境與應用選擇適當的Search Strategy。以下介紹幾種Search Strategy並分析其優缺點。

[Full Search]

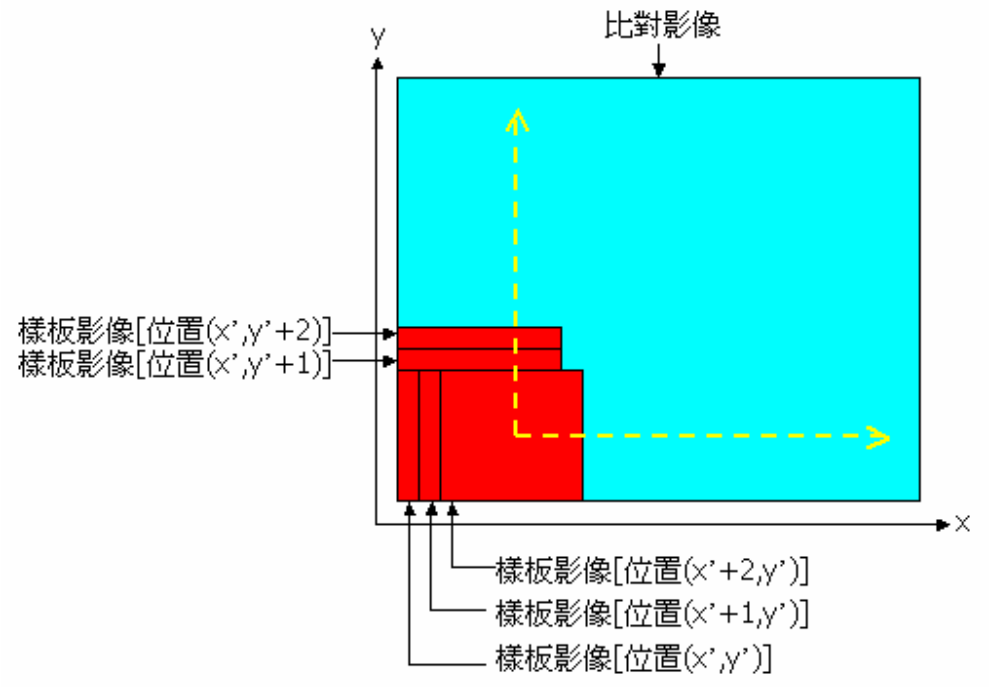


圖 5-3-1 Full Search

圖 5-3-1 為利用 Full Search 的方式進行樣板比對，藍色影像為比對影像、紅色影像為樣板影像。比對方式從最左下的樣板影像[樣板中心點座標(x',y')]開始進行比對，接著移動樣板 x 軸 1 個像素距離或移動 y 軸 1 個像素距離，逐步的進行比對的工作，以這樣的 Search 方式，找出最相似的影像子區塊。Full Search 是以逐步增加 x 軸或 y 軸像素距離去進行比對的工作，因此比對的結果較準確，但是因為它針對影像中每一位置逐一比對，所耗費的計算量及計算時間都太大。

[N-Pixel-Shifting Search]

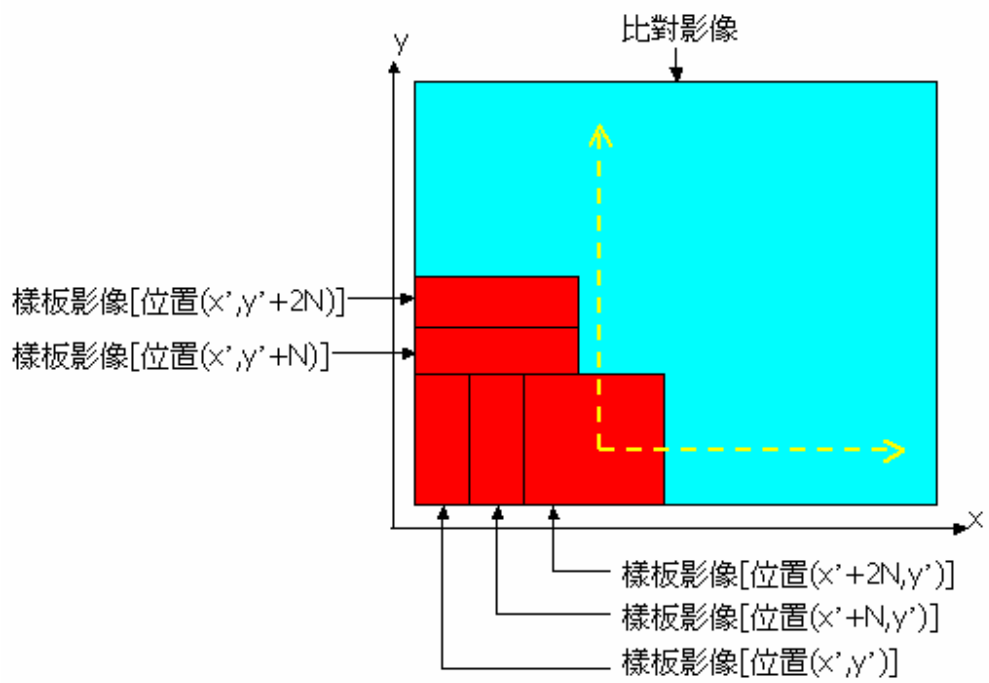


圖 5-3-2 N-Pixel-Shifting Search

圖 5-3-2 為 N-Pixel-Shifting Search 示意圖，此 Search 方式和 Full Search 方式唯一的不同是，Full Search 方式是移動 x 軸或 y 軸一個像素的距離來進行比對的動作。N-Pixel-Shifting Search 方式是以移動 x 軸或 y 軸 N 個像素距離的方式進行比對的動作。此 Search 方式因為是以 N 個像素距離為移動的單位，因此比對時的所花費的計算量及計算時間相較於 Full Search 小。但因移動間隔為 N 個像素距離，因此樣板比對的精確度較小。

[Down-Sampling Search]

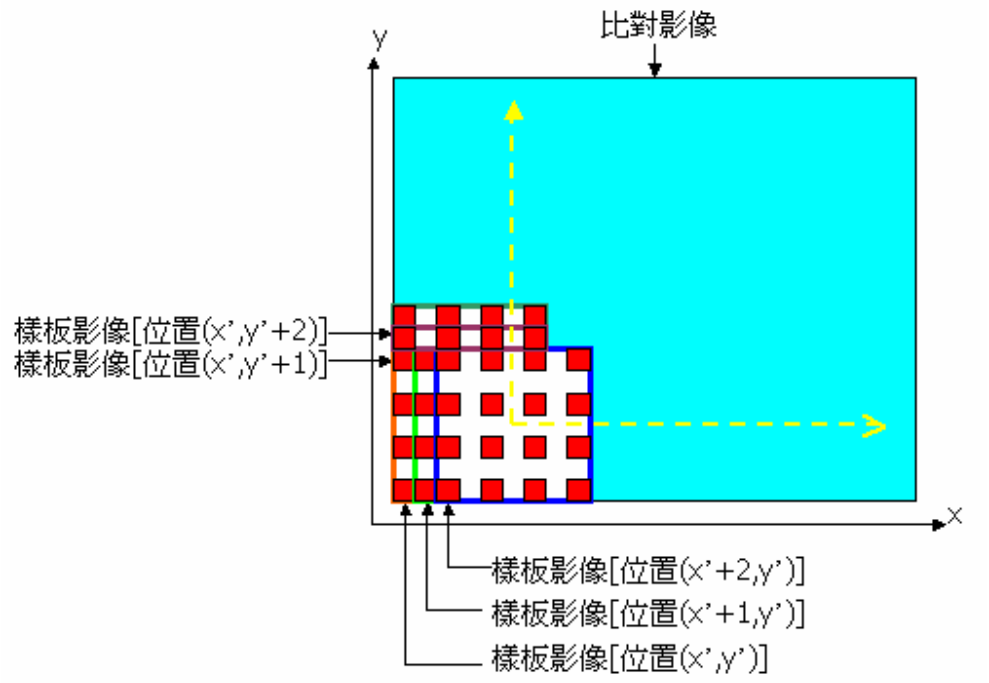


圖 5-3-3 Down-Sampling Search

圖 5-3-3 為 Down-Sampling Search 的示意圖，在進行比對時，樣板移動的方式和 Full Search 是一樣的，以一個像素距離為移動的單位，進行 x 軸、y 軸方向的比對。但在進行相似性 metric(difference、cross-correlation)的計算時，並沒有比對全部的樣板影像，而是比對樣板經 down-sampling 後的影像。以下以圖 5-3-4 說明比對的方式：

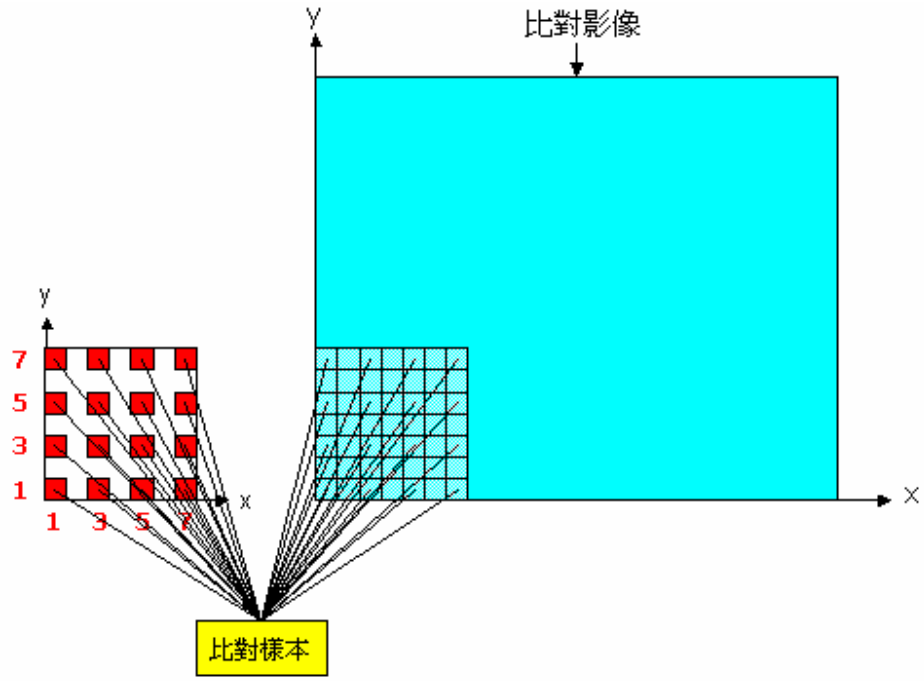


圖 5-3-4 比對樣本

我們以圖 5-3-4 來說明 down-sampling 比對的計算方法及其比對的樣本，並以最左下角的影像子區塊來說明。如圖所示，拿出樣板影像和比對影像的 x 軸、y 軸的 1、3、5、7 位置的像素，比對二者相對位置像素的 difference 或 cross-correlation，而 x、y 座標為 2、4、6 的像素是不處理的。

由上可知 Down-Sampling Search 的運算方式與流程，因其比對樣本為原始樣板的某些取樣點，因此要處理的像素變少、計算量較少。而在準確度方面，因其使用 sampling 的樣本，準確度比 Full Search 方式低。

[Coarse-to-Fine Search]

經由上述的介紹，我們知其個別的 Search 方式都有其優缺點，有些為精確度高，但計算量卻很高，有些則計算量低，但其精確度卻較低。正因為有計算量、精確度二者取捨的考量，因此可依序使用二種 Search 的方式，使其計算量及精確度都在一可接受的範圍之內，我們稱這種 Search Strategy 為 Coarse-to-Fine Search，以下說明此 Search 方式的運作流程：

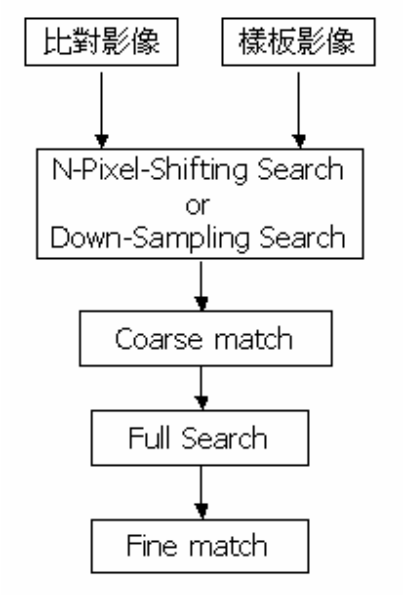


圖 5-3-5 Coarse-to-Fine Search 流程圖

圖 5-3-5 為 Coarse-to-Fine Search 流程圖，首先給定比對影像和樣板影像的資訊，接著使用 N-Pixel-Shifting Search 或 Down-Sampling Search 方式，快速的比對樣板和輸入影像最相似的地方，雖然經此樣板比對之後的結果可能不是非常的準確，但其實已經粗略的定位出樣板和

比對影像的對應位置了。知道粗略的位置資訊之後，可進一步的利用 Full Search 方式，搜尋前述所定位之位置的附近區域，精確的找出樣板和輸入影像中最相近的子區塊。

5.4 手勢辨識

在此章節裡，我們將提到物件辨識相關的影像處理，其中又特別針對手勢辨識的部分加以探討。針對不同的手勢樣式，要能分辨出各種手勢樣式的不同，此動作稱為〝手勢辨識〞。以下將說明在進行手勢辨識時，相關的運算流程及基本原理。

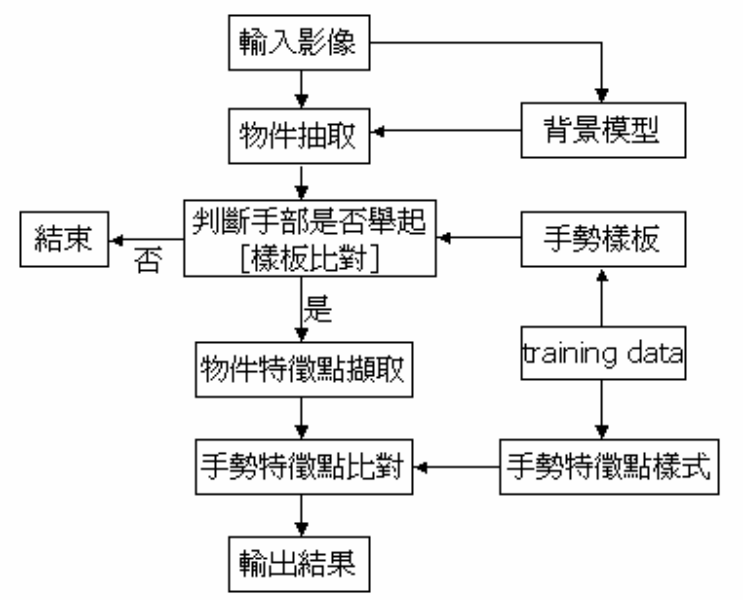


圖 5-4-1 手勢辨識運算流程

圖 5-4-1 列出了整個手勢辨識的運算流程，我們先介紹整個手勢辨

識的運算流程圖，先概略的知道其運算流程，等有了基本概念之後，再去細看各子區塊的運算。

首先針對輸入影像進行物件抽取的動作。接著再利用預先 training 的手勢樣板資料，和剛才抽取出的物件影像進行樣板比對，依比對之後的 difference 或 cross-correlation 的值判斷此人手部是舉起。如果判斷出此人手部沒有舉起，則結束後續運算；反之，如判斷出此人手部舉起，則進行後續運算。對剛才經樣板比對出的手部位置區塊，進行特徵點的擷取動作，將這些手部特徵點的分佈資訊，和預先 training 的手勢特徵點樣式進行手勢特徵點的比對動作，經特徵點比對之後，便可得最符合的手勢樣式。經由手勢辨識運算流程圖及上述相關說明後，可概念性的知道各手勢是如何進行辨識的工作，上面為概念性的說明，以下將會針對上述各流程區塊的運算加以說明。依序說明[物件抽取]、[樣板比對]、[物件特徵點擷取]、[手勢特徵點比對]各步驟的運算。

[物件抽取]

一般的影像大都可由前景影像及背景影像組成，物件抽取則是一種將影像的前景部分抽取出來的技術。前景影像即是影像中和背景影像不同的地方，因此在進行物件抽取時，需先建立背景影像，以利前景影像的取得。建立背景模型的方法有很多種，常見的有Average/Median

Method及Running Average Method[10]、[11]。用Average/Median Method建立背景模型的方法是觀察連續N張影像，藉由這N張影像的資料，找出每個像素位置的平均值及中間值做為背景影像，數學式如下：

$$B_{i+1}(x,y)=\frac{B_i(x,y)+B_{i-1}(x,y)+...B_{i-N+1}(x,y)}{N}\ for\ \ all\ \ (x,y)\ \ (Average\ method) \quad (5.3)$$

$$B_{i+1}(x,y)=median\{B_i(x,y),B_{i-1}(x,y),...B_{i-N+1}(x,y)\}\ for\ \ all\ \ (x,y)\ \ (Median\ method) \quad (5.4)$$

如上數學式(5.3)、(5.4)所示，某一時間點的背景模型，需要利用到前N個影像畫面的資訊。針對每一座標的像素值(Pixel Value)，Average Method的方式是將前面N張影像畫面取其平均值，Median Method的方式是N個像素值進行由大到小的排列，取其中間值。上述二方法有一個很大缺點，即需要浪費大量的記憶體。

另一種常見的背景模型為使用Running Average Method，數學式如下數學式(5.5)所示：

$$B_{i+1}=\alpha*F_i+(1-\alpha)*B_i \quad (5.5)$$

其中F為前景、B為背景、α為學習速率(Learning rate)。此方法是藉由每一影像畫面(F_i)的更新，逐步調整其背景(B_{i+1})。

當背景模型建立好之後，便可利用背景相減法進行物件抽取的動作，背景相減法的運算式如下[12]：

$$IF \quad \left| frame_{(x,y)} - background_{(x,y)} \right| > Th$$

$$THEN \quad foreground_{(x,y)} = frame_{(x,y)}$$

$$ELSE \quad foreground_{(x,y)} = 0 \tag{5.6}$$

上述數學式 (5.6)中的 $frame_{(x,y)}$ 為原始影像中在座標(x,y)的像素值，$background_{(x,y)}$ 為背景影像在座標(x,y)的像素值， $foreground_{(x,y)}$ 為前景影像在座標(x,y)的像素值，Th 為某一給定的門檻值。

經由背景相減法之後，雖然可取得前景影像，但是此時的前景影像是很粗糙的。除了包含雜訊之外，針對每一物件區域裡可能有因為切割的不完全，以致在物件區域裡有漏洞(hole)的情形。以上因為物件切割的不完美，所造成前景影像的不完整，都要經過影像後置處理來修補。針對小區塊及突波雜訊部分，可利用Opening的運算來消除雜訊[13]。另外針對每一區域內如有漏洞的話，可利用Closing運算來補齊漏洞[13]。上述說明為整個物件抽取的說明，以下列出完整的物件抽取流程圖，如圖5-4-2所示：

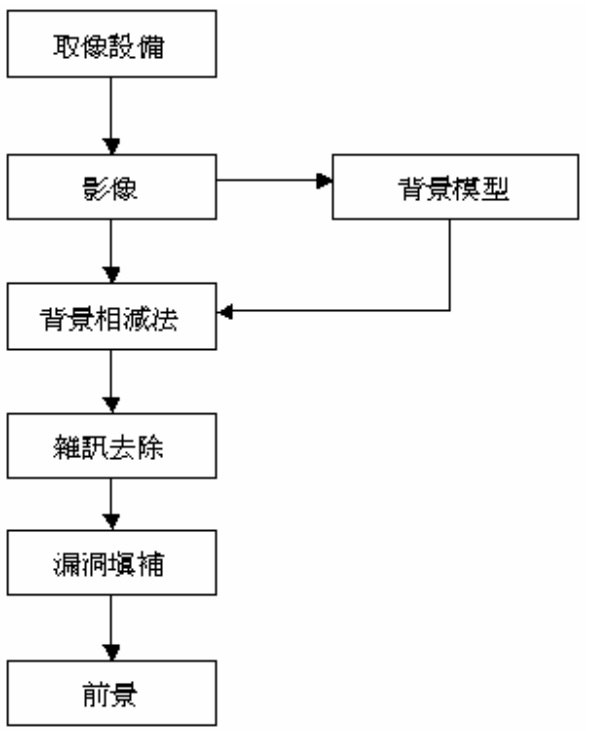


圖5-4-2 物件抽取流程圖

圖 5-4-2 為整個物件抽取的流程圖，從前端利用取像設備取得影像資料，然後藉著輸入的影像資料建立背景模型。接著利用背景相減法取得粗糙的前景影像，再來便將此粗糙影像送到影像的後置處理部分，進行雜訊去除、漏洞填補等，使其前景影像較完整、正確。以下以一例子顯示物件抽取的運算結果：

圖 5-4-3 原始影像(包含前景與背景)

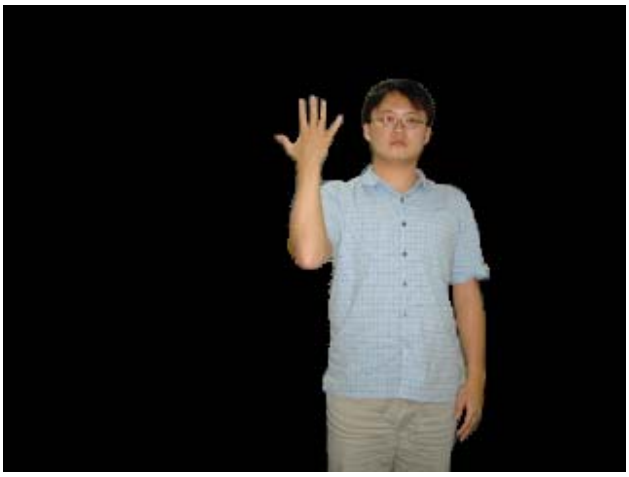


圖 5-4-4 物件影像(前景)

圖 5-4-3 為原始影像，此影像是由前景影像及背景影像所組成。圖 5-4-4 是圖 5-4-3 經由物件切割運算後，所求得的物件影像。

[樣板比對]

樣板比對的相關技術與原理在章節 5.3 裡有完整的運算流程與說明，在此僅說明手勢辨識相關的樣板比對技術。然而因為我們要進行的是手勢的樣板比對，因此要事先定義手勢樣本，在此定義了五個手勢，手勢樣本如圖 5-4-5 所示。

圖 5-4-5 手勢樣本

有了圖 5-4-5 的手勢樣本之後，便可以利用此手勢樣本和物件抽取後的物件影像進行樣板比對的動作，比對完之後，便可依上述二者的 difference 或 cross-correlation 的值判斷比對者的手部是否舉起。判斷的結果如手部沒有舉起，則結束手勢辨識動作；如手部舉起，則繼續進行手勢辨識的後續動作。

雖然樣板比對的技術在章節 5.3 已有說明其運算方式，但是在此處的比對方式和前述有所不同。不同之處在於原始樣板比對處理的對象為整張完整影像，此處僅針對影像中前景部分加以處理。首先找出框出前景的矩形框，此矩形框內的範圍為樣本比對的 Search 範圍，此作法可

減少樣板比對時 Search 的範圍。更進一步地，我們設定手部舉起時，需有合理的位置分佈，只針對合理的手部位置進行比對，如此又可減少樣板比對的 Search 範圍，上述說明的 Search Strategy 以圖 5-4-6 來呈現。

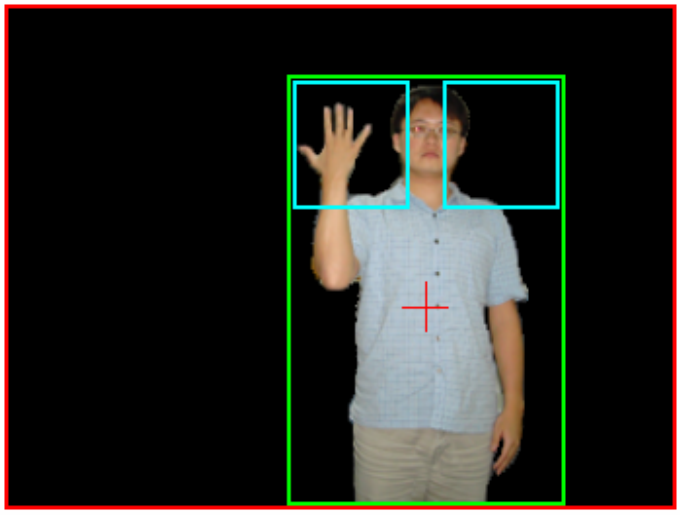


圖 5-4-6 樣板比對範圍

我們藉由圖 5-4-6 來說明樣板比對時所處理的影像範圍，傳統樣板比對的處理範圍為整張影像，即紅色矩形框內範圍，因此比對範圍大、計算量大。進一步地，如果只針對前景影像處理的話，便可減少比對的範圍，方法如下：首先計算前景影像中非零值像素的總像素及物件中心點位置，依中心點位置及總像素的大小框出前景物件，即綠色矩形框，此時樣板比對範圍較上述範圍小、計算量小。如果我們再進一步搭配手部舉起時，合理的位置分佈，可進一步的縮小樣板比對範圍。上述所說

之合理的手部位置資訊，即定義在物件中心點的左、右上方區塊。以圖 5-4-6 為例，即兩藍色矩形框區域。樣板比對的 Search 範圍由原本的整張影像縮小到前景，進一步的縮小到合理的手部分佈範圍，大大的減少比對時所需的計算量及計算時間。樣本比對範圍經由上述步驟定義出之後，便可利用 Coarse-to-Fine Search 方式的樣板比對，找出手部分佈的位置。

[物件特徵點擷取]

如經樣板比對後，判斷出手部舉起時，便繼續進行物件特徵點的擷取動作。此處所執行的特徵點擷取動作，是利用 Harris Corner Extraction Algorithm 所完成的，此演算法的運算流程在章節 5.2 有詳盡的說明。

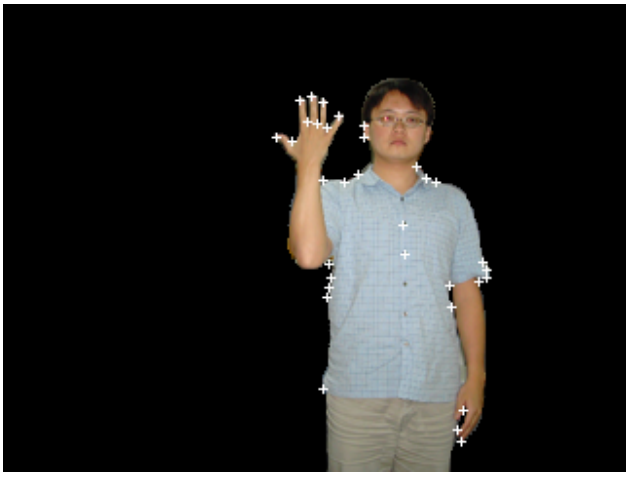


圖 5-4-7 物件特徵點分佈

圖 5-4-7 為圖 5-4-4(物件影像)進行 Harris Corner Extraction 運算之後，所求得的特徵點分佈圖，以十字記號標記 Corner 的位置。當我們得到物件的特徵點分佈圖之後，便可以搭配手勢特徵點樣式及手部位置資訊，進行手勢特徵點比對的動作。

[手勢特徵點樣式]

當我們完成物件特徵點擷取的工作之後，接著要進行的運算流程便是手勢特徵點的比對，藉由特徵點的比對結果，判斷出相對的手勢樣本，於是便完成手勢辨識的運算流程。但在進行手勢特徵點比對之前，需事先知道手勢特徵點樣式的資訊，以下將介紹手勢特徵點樣式的取得方式及相關定義。以圖 5-4-8 來說明：

(a)

(b)

(c)

(d)

(e)

圖 5-4-8 手勢特徵點樣式

藉由圖 5-4-8(a)~(e)的圖形演進，說明手勢特徵點樣式的前置處理過程。以下將說明各圖形是經由何種運算得來的：

(1)圖 5-4-8(a)是我們事先定義的手勢樣本，由左到右依序為手勢 5、手勢 4、手勢 3、手勢 2、手勢 1。

(2)圖 5-4-8(b)、(c)為圖 5-4-8(a)的手勢樣本經特徵點擷取運算後，再針對特徵點影像進行 thresholding 動作後，所求得特徵點分佈。

圖 5-4-8(b)是利用低門檻值進行 thresholding 動作所得的特徵點分佈，圖 5-4-8(c)是利用高門檻值進行 thresholding 動作所得的特徵點分佈。

(3)圖 5-4-8(d)是由圖 5-4-8(c)的特徵點分佈資訊，進行特徵點分群動作所求得的影像。以下以一簡單例子說明特徵點分群：

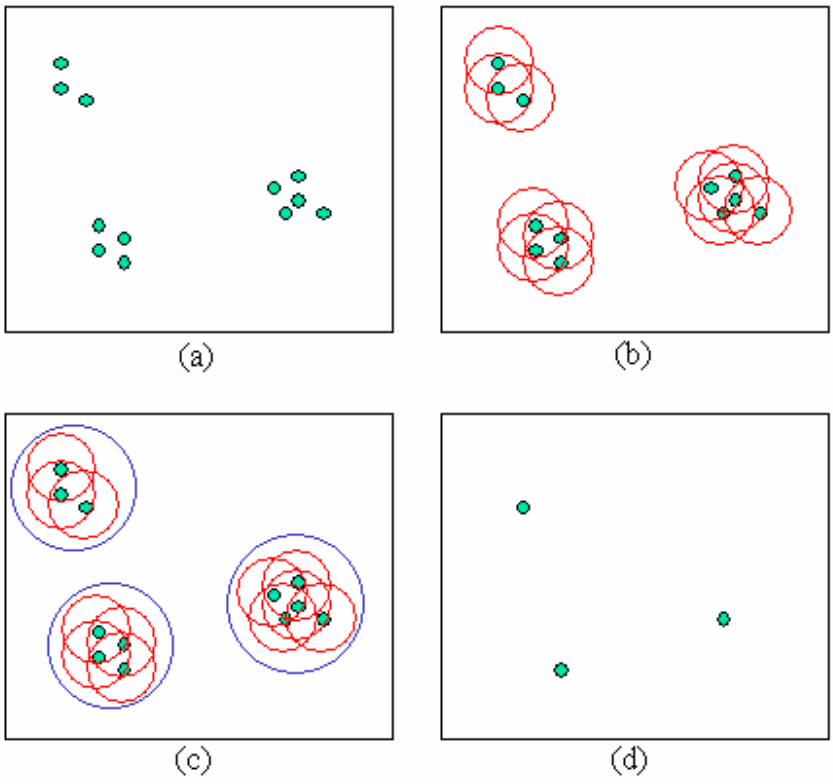


圖 5-4-9 特徵點分群

我們以圖 5-4-9 來說明特徵點分群的動作，圖 5-4-9(a)為某一影像的特徵點分佈情形。圖 5-4-9(b)為將每一特徵點，以自身的位置為圓心，某一半徑的值畫出一圓。針對某一個特徵點，如畫出的圓包含另一特徵點，則此二點屬於同一群體，針對每一特徵點反覆的進行群體分類

的動作，最後可將分佈相近的特徵點歸類為某一群體，如圖 5-4-9(c)為將所有的特徵點分為三個群體。針對每一群體，任意的選取群體內的某一特徵點，將其保留下來，其他的特徵點予以丟棄，如此例有 3 個群體，保留下的特徵點為 3 點，其結果如圖 5-4-9(d)所示。

(4)圖 5-4-8(e)為最終的特徵點樣式分佈圖，是利用圖 5-4-8(d)影像，保留指尖附近的特徵點，其餘距離指離指尖較遠的特徵點予以丟棄，所得的特徵點樣式影像。以圖 5-4-10 說明：

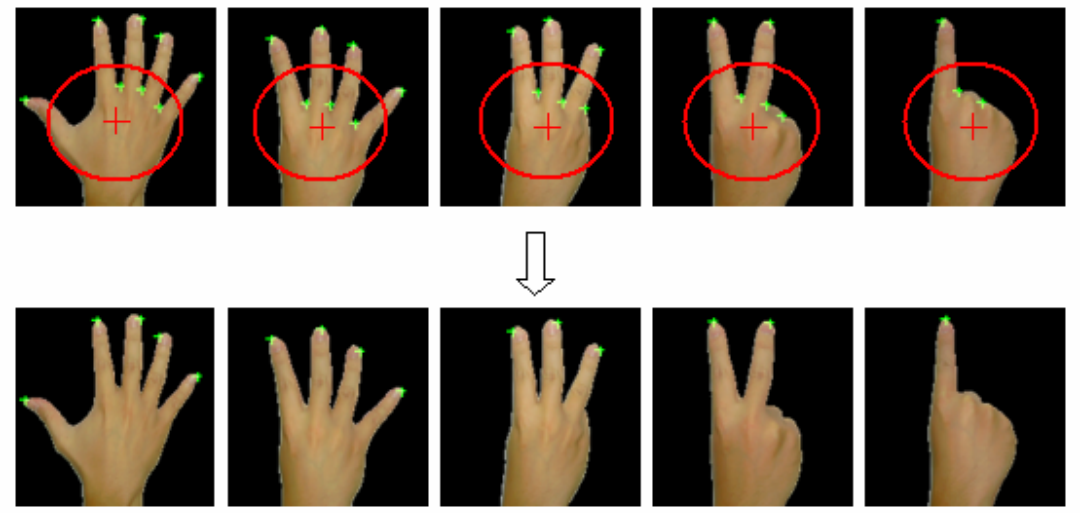


圖 5-4-10 特徵點樣式求法

我們藉由圖 5-4-10 來說明最後的特徵點樣式是如何求得的。首先針對每一個手勢樣本找其中心點的位置，再依手勢樣本的大小及中心點的位置畫出一適當的圓。特徵點如分佈在圓外則保留特徵點，在圓內則將其捨棄，以此方式可得最後的手勢特徵點樣式。

[手勢特徵點比對]

當物件特徵點及手勢特徵點樣式都已經藉由上述運算流程得到之後，便可以著手進行手勢特徵點比對的動作，以下介紹手勢特徵點的比對方式，以圖 5-4-11 為例。

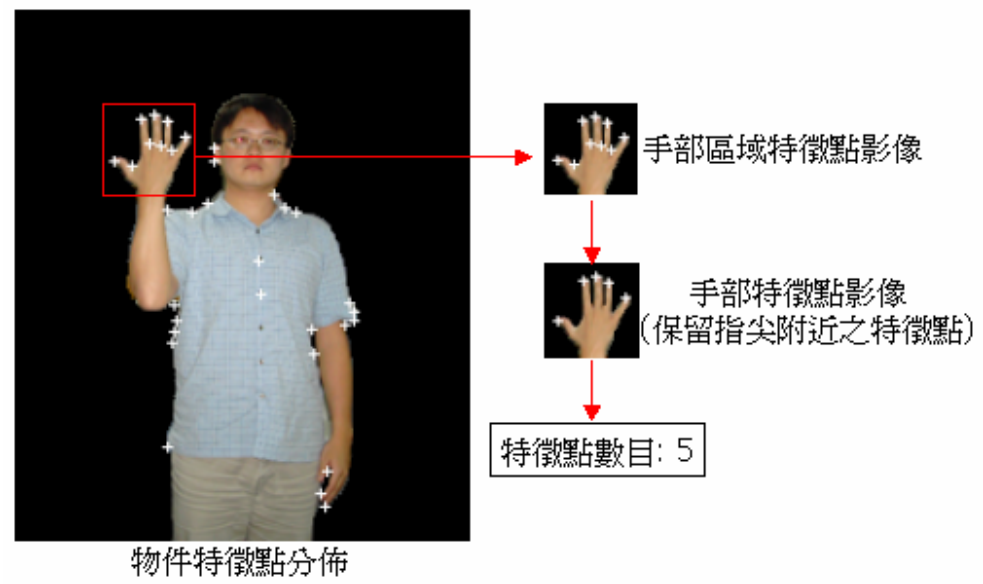


圖 5-4-11 特徵點比對

首先對物件特徵點影像中，取出手部位置的特徵點影像，接著捨棄不重要的特徵點資訊，保留指尖附近的特徵點資訊，接著計算此時的特徵點數目，此時比對各手勢特徵點樣式的特徵點數目，二者一比較便可知此時的手勢資訊。以此例而言，計算出的特徵點數目為 5，比對各手勢的特徵點數目，可得此手勢為手勢 5。

第六章 系統實現

6.1 簡介

上述章節的討論都只著重在整個即時互動式監控系統所使用的傳輸技術、互動機制、物件定位與手勢辨識的原理及說明，而在此章節將會介紹系統的軟體實現部分，依序介紹Client/Server端的使用者介面、Client/Server端架構、Client端程式運作流程、Client端之各Thread間的互動關係。

6.2 Client/Server 端使用者介面

下面將會介紹我們所佈建的監控平台的 Client 端及 Server 端的使用者介面，並介紹二介面所提供的功能。

圖 6-2-1 Server 端介面

圖 6-2-1 為監控平台 Server 端的使用者介面，整個介面有幾個較重要的元件區塊要注意，分別為取像相機設定區、連線狀態顯示區、傳輸影像顯示區、錄影設定區。以下將依序介紹各區塊的功用：

[取像相機設定區]

此區域可設定 Server 端所要收集的相機影像，使用者需輸入欲取像之 IP Camera 的 IP Address。

[連線狀態顯示區]

此區域會有針對各 Client 和 Server 所建立的 RTSP Connection 顯示其連線狀態。諸如連線的數目、各連線的資料傳輸速率等等。

[傳輸影像顯示區]

此區域會顯示 Server 端要傳送給 Client 端的影像。

[錄影設定區]

此區域可設定錄影的動作。舉例來說，如使用者勾選 REC Checkbox 的話，Server 會針對此時傳輸給 Client 的影像進行錄影的動作。反之，如勾選 REC_END Checkbox 的話，會結束錄影的動作。

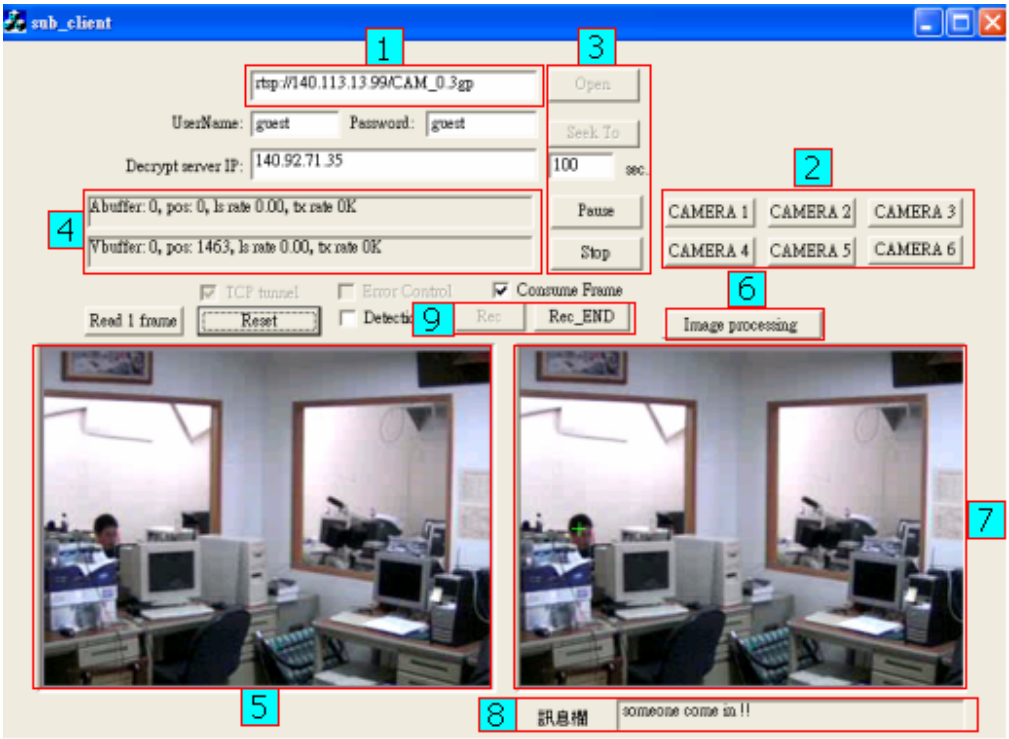


圖 6-2-2 Client 端介面

圖 6-2-2 為監控平台 Client 端的使用者介面，以下將依序介紹各區域的功用：

[區域 1]

指定欲連線的 Media Server 位址。

[區域 2]

設定欲觀看的相機影像。

[區域 3]

OPEN：建立 RTSP Connection 並發出 PLAY Message 給 Server，接著等

待 Server 端傳送 video/audio 資料。

SeeK To：如使用者觀看的 Streaming data 為 stored data，此時可利用此按鈕及所屬的 Edit 輸入區，設定欲播放的影像檔時間。

Pause：如使用者觀看的 Streaming data 為 stored data，此時可利用此按鈕暫停影像檔的播放。

Stop：結束影像檔案播放的動作且關閉此 RTSP Connection，並釋放相關資源。

[區域 4]

此區域顯示 Client 端接收 Server 端所傳送的 Video/Audio 資料的接收情形。

[區域 5]

Client 端收到 Streaming data 之後，會對此資料進行解碼的動作，經解碼之後所得到的 RGB 影像畫面即顯示在此區域。

[區域 6]

Image processing：Client 端如要針對收到的影像畫面進行影像處理的話，可按下此按鈕進行影像處理的動作。

[區域 7]

此區域顯示影像處理後的輸出影像。

[區域 8]

此處顯示影像處理後的輸出訊息欄。

[區域 9]

當 Client 端的使用者想要針對此時的影像進行錄影的動作的話，可按下按鈕"Rec"告知 Server 進行錄影；欲結束錄影則按下按鈕"Rec_END"告知 Server 結束錄影。

6.3 Client/Server 端架構

[Server 端架構]

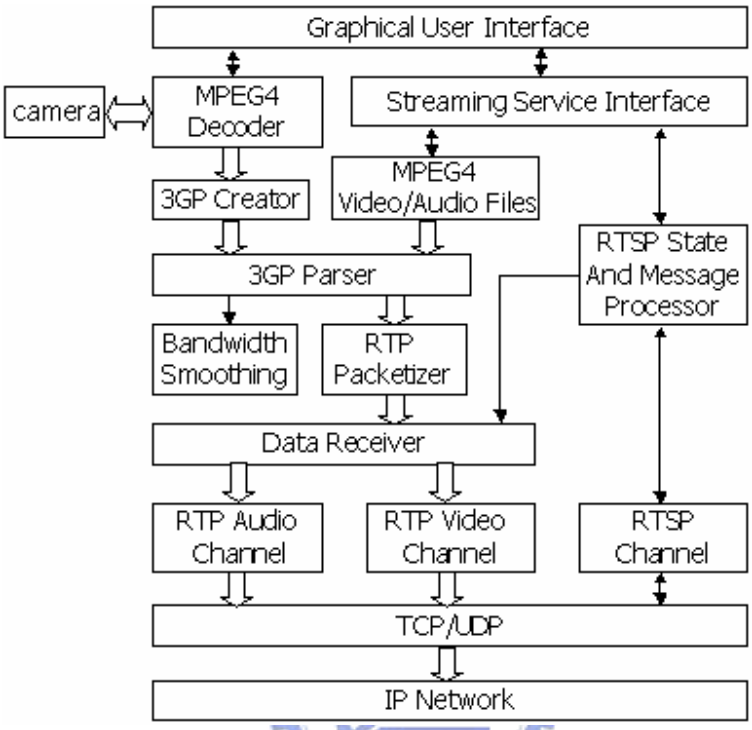


圖 6-3-1 Server 端架構

圖 6-3-1 為我們所佈建的監控平台的 Server 端架構，利用此架構實現具串流傳輸的互動式監控平台。以下將介紹各區塊功能及系統運作流程。首先，Graphical User Interface(GUI)為電腦程式和使用者之間的介面，使用者可按下介面上的 button 啟動相對應的程式運算動作。舉例來說，如使用者按下〝streaming start〞button，此時 GUI 接收到〝streaming start〞button 按下的指令，會執行此 button 對應的動作。首先會透過 Streaming Service Interface 建立 Streaming server

的角色以提供 Client 連線。如有 Client 端向 Server 發出建立連線訊息的話，Server 端會透過 RTSP Channel 收到此連線要求訊息，接著將此訊息告知 RTSP State and Message Processor 處理此訊息。接著此 Processor 會透過 Streaming Service Interface 針對 Client 端所要求的影音檔案進行處理。分別針對 stored video、lived capture 的影像透過 MPEG4 Encoder、3GP Parser、RTP Packetizer、Data Streamer 等裝置將影音檔案包裝成 RTP Audio、RTP Video 封包，透過 RTP Audio、RTP Video Channel 將封包傳輸出去，而 Client 端將透過網路接收到此影音資料。而 Client 端如果想針對此連線所取得的影音檔進行暫停、停止、錄影等動作的話，只要再透過對應的 RTSP Message 傳送給 Server 端告知欲進行的動作即可。Server 端經 RTSP Channel 取得 RTSP Message 之後，交由 RTSP State and Message Processor 處理，此 Processor 會依所掌握的 Client 端狀態及 RTSP Message，執行相對應的動作，如此一來，Client/Server 二端可達到互動式的功能。

[Client 端架構]

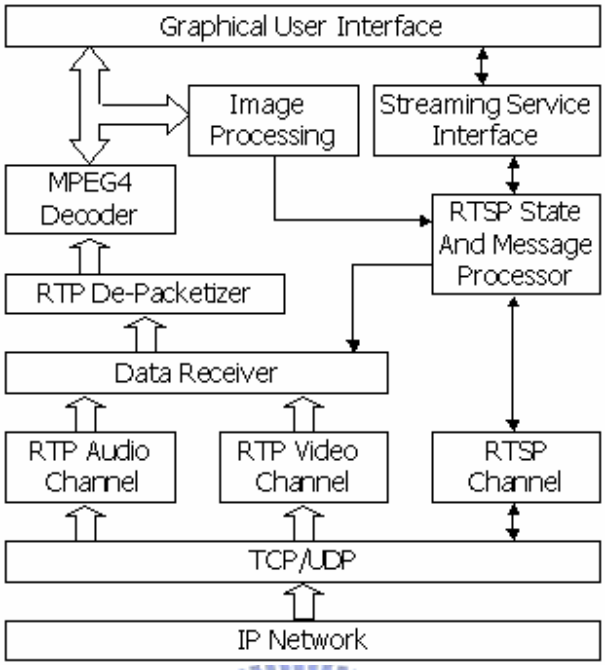


圖 6-3-2 Client 端架構

圖 6-3-2 為我們所佈建的監控平台的 Client 端架構，以下將介紹各區塊功能及系統運作流程。首先，Graphical User Interface(GUI)為電腦程式和使用者之間的介面，使用者可按下介面上的 button 啟動相對應的程式運算動作。舉例來說，如使用者按下〝OPEN〞button，此時 GUI 接收到〝OPEN〞button 按下的指令，會執行此 button 對應的動作。首先會透過 Streaming Service Interface 和 Server 端建立連線。連線的建立是透過 RTSP State and Message Processor 透過 RTSP Channel 發出建立連線要求的 RTSP Message 給 Server。當 Server 端接收到此 RTSP Message 時，如同意此連線要求的話，會將收集到的影音資料透過 RTP Audio、RTP Video Channel 傳送給 Client 端。而 Client

端也會透過 RTP Audio、RTP Video Channel 接收到此影像資料，接著透過 RTP De-Packetizer 取出 MPEG4 影像區段，再經由 MPEG4 Decoder 解碼成 RGB 資料型式的影像。接著將影像送到 GUI 呈現及交給 Image Processing 程式處理，將影像處理後的結果對應事先定義的〝事件定義〞之後，便知道該執行何種後續動作。舉例來說，如處理後的結果為〝入侵者進行〞，此時可能要告知 Server 端進行錄影的動作。Client 端是透過 RTSP State and Message Processor 發出夾帶錄影訊息字串的 RTSP Message 給 Server 端。當 Server 端經由 RTSP Channel 收到此夾帶錄影訊息字串的 RTSP Message 時，便知道 Client 要求錄影的動作，接著依此錄影訊息字串執行錄影的動作。

6.4 Client 端程式運作流程

在章節 6.2 裡，介紹了監控平台的 Client/Server 端的使用者介面及相關的使用說明。進一步地，將介紹這些使用者介面是如何透過相關軟體程式達到所提供的功能。下面將介紹 Client 端的程式運作流程：

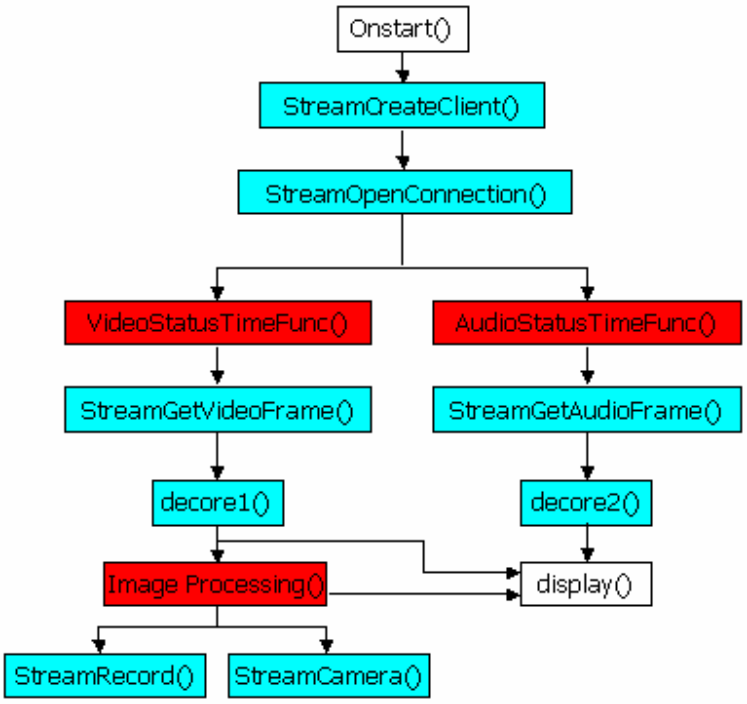


圖 6-4-1 Client 端程式運作流程

圖 6-4-1 為 Client 端的程式運作流程，描述著 Client 端主要的程式運作。以下將依序介紹各程式區塊的功用：

當 Client 端的使用者按下 button 〝Open〞時，會執行 Member Function Onstart()內的程式碼。首先會透過 StreamCreateClient()函式建構出建立 RTSP Connection 所需的物件，接著會利用 StreamOpenConnection()建立 RTSP Connection 並發出 PLAY Method 給

Server，告知欲進行影像檔案播放的動作，接著等待 Server 端傳送過來的 Video/Audio data。

接著會開啓 VideoStatusTimeFunc()、AudioStatusTimeFunc() Thread 負責處理 Video 及 Audio 資料。二 Thread 會分別呼叫 StreamGetVideoFrame()、StreamGetAudioFrame()函式取得 Client 端 Buffer 內的 Video 及 Audio 資料，接著再透過個別的解碼器 decore1()、decore2()解碼出 RGB 影像及聲音訊號。接著會將 RGB 影像交給 Image Processing() Thread 進行影像處理的動作，搭配事先所定義的錄影機制及相機切換機制，根據影像處理的結果決定是否要執行錄影的動作或相機的切換。如要錄影可呼叫 StreamRecord()函式，此函式會藉由 PING Message 的使用來告知 Server 端進行錄影的動作。如要進行相機的切換動作時，可呼叫 StreamCamera()函式告知 Server 端進行相機切換的動作。

上述為整個 Client 端程式運算流程的簡短說明，以下將針對主要的程式區塊進行詳細的說明：

[StreamOpenConnection]

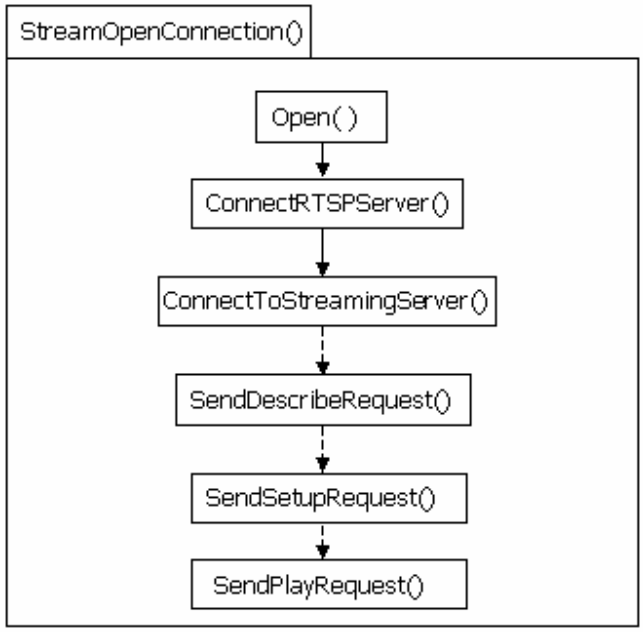


圖 6-4-2 StreamOpenConnection()程式流程

圖 6-4-2 為 StreamOpenConnection()函式的內部程式運作流程圖。函式一開始會執行 open()函式，此函式會呼叫 ConnectRTSPServer() 函式，ConnectRTSPServer()會透過 ConnectToStreamingServer()函式建立及測試 Client/Server 二端的 TCP Connection 是否連線正常。如 TCP Connection 連線正常的話，Client 端會藉由 SendDescribeRequest()函式發出 RTSP 的 Describe Message 給 Server 端以取得取得 Media Data 的相關描述。有了 Media Data 的相關訊息之後，便知道 Media Server 所處位置。接著透過 SendSetupRequest()函式發出 Setup Message 和

Server 端進行連線的設定動作。接著透過 SendPlayRequest()函式發出 Play Message 給 Server 端，告知欲進行影像檔案的播放動作。

[StreamGetVideoFrame、decore1]

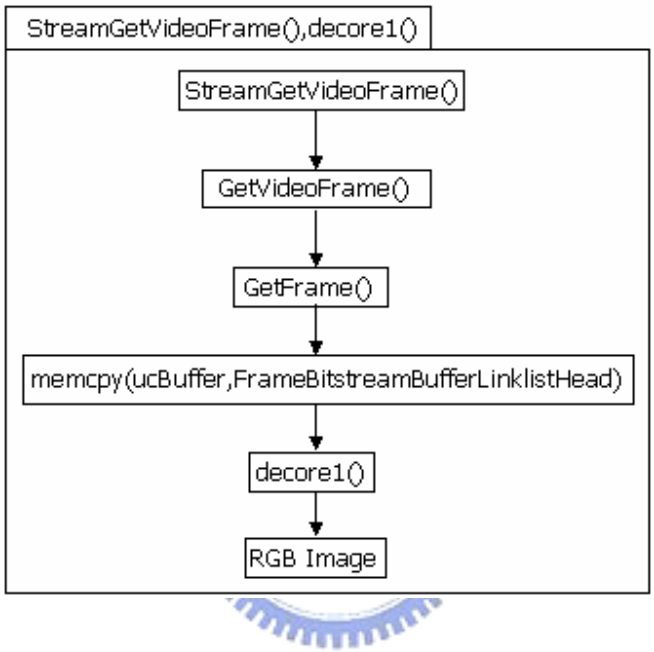


圖 6-4-3 StreamGetVideoFrame、decore1 程式流程

圖 6-4-3 為 StreamGetVideoFrame、decore1 函式的程式運作流程圖。首先，StreamGetVideoFrame 函式會呼叫 GetVideoFrame()，GetVideoFrame()函式會進一步的呼叫 GetFrame()。GetFrame()函式會將 Client 端收到的 RTP Packet 組合成 MP4 資料並儲存於 FrameBitstreamBufferLinklistHead Buffer 裡，接著會進行 memory copy 的動作，將此 Buffer 內的 MP4 資料複製到所指定 ucBuffer 裡。當

有了 MP4 檔案之後，便可利用 MP4 Decoder decore1()進行解碼的動作，解碼後的檔案格式為 RGB Format 的影像資料。

[StreamRecord]

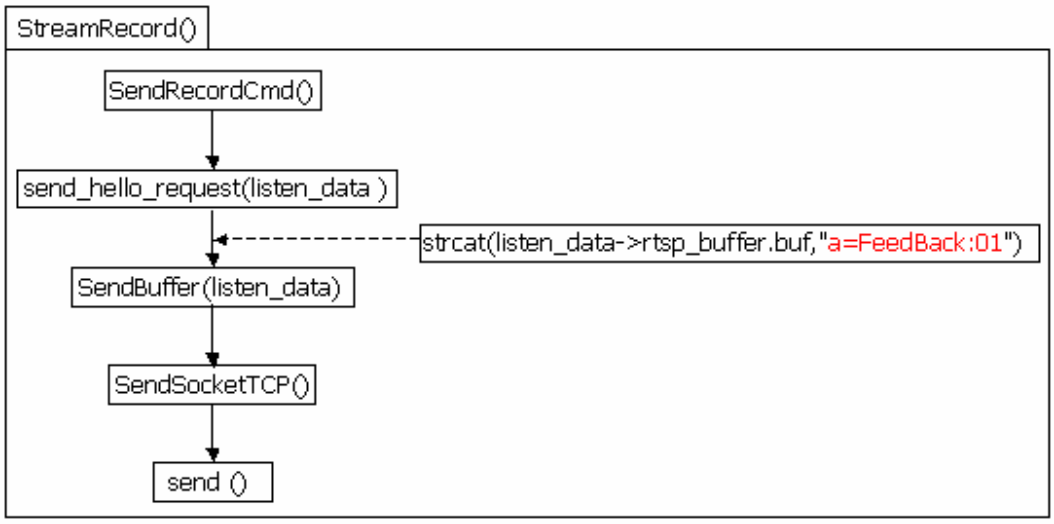


圖 6-4-4 StreamRecord 程式流程

當我們針對影像資料進行影像處理的運算時，處理後的結果可能會符合事先定義的事件情況，因此可能要對此時的影像進行錄影的動作，此時可呼叫 StreamRecord()函式通知 Server 端告知錄影的動作，詳細程式運作流程如圖 6-4-4 所示。

StreamRecord() 函式會先呼叫 SendRecordCmd() 函式，SendRecordCmd()函式會再去呼叫 send_hello_request()函式，接著將錄影訊息字串〝a=FeedBack:01〞塞入 PING Message 裡，PING Message 包裝好之後，交給 SendBuffer() 函式，SendBuffer() 再呼叫

SendSocketTCP()函式，SendSocketTCP()再透過 Send()函式將此 PING Message 送到 TCP Socket 傳輸。當 Server 端經由 TCP Socket 接收到此包含錄影訊息字串的 PING Message 時，會知道 Client 端要求錄影的動作，並執行錄影動作。

[StreamCamera]

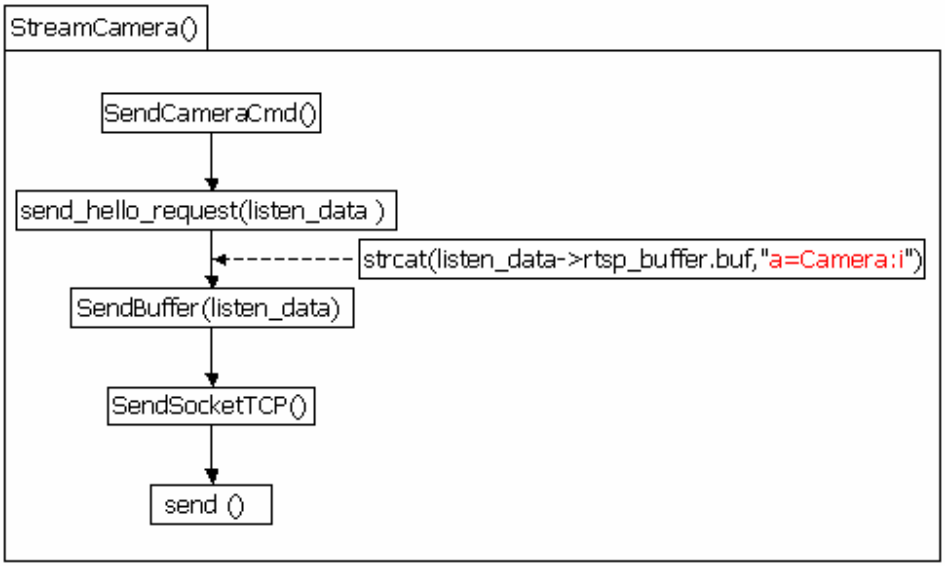


圖 6-4-5 StreamCamera 程式流程

當我們對影像資料進行影像處理的運算時，有時可能會因某種應用的要求而需要有不同的相機影像，因此 Server 端傳送給 Client 端的相機影像需要適時的切換。因此 Client 端可利用 StreamCamera()函式告知 Server 端進行相機的切換動作。詳細程式運作流程如圖 6-3-5 所示。

StreamRecord() 函式會先呼叫 SendCameraCmd() 函式，SendCameraCmd()函式會再去呼叫 send_hello_request()函式，接著將 Client 端想要切換的相機訊息字串〝a=Camera:i〞塞入 PING Message 裡，PING Message 包裝好之後，交給 SendBuffer()函式，SendBuffer() 再呼叫 SendSocketTCP()函式，SendSocketTCP()再透過 Send()函式將此 PING Message 送到 TCP Socket 傳輸。當 Server 端經由 TCP Socket 接收到此包含相機訊息字串的 PING Message 時，會依所隱藏的相機訊息切換傳送給 Client 端的相機影像。

6.5 Client 端之各 Thread 間的互動關係

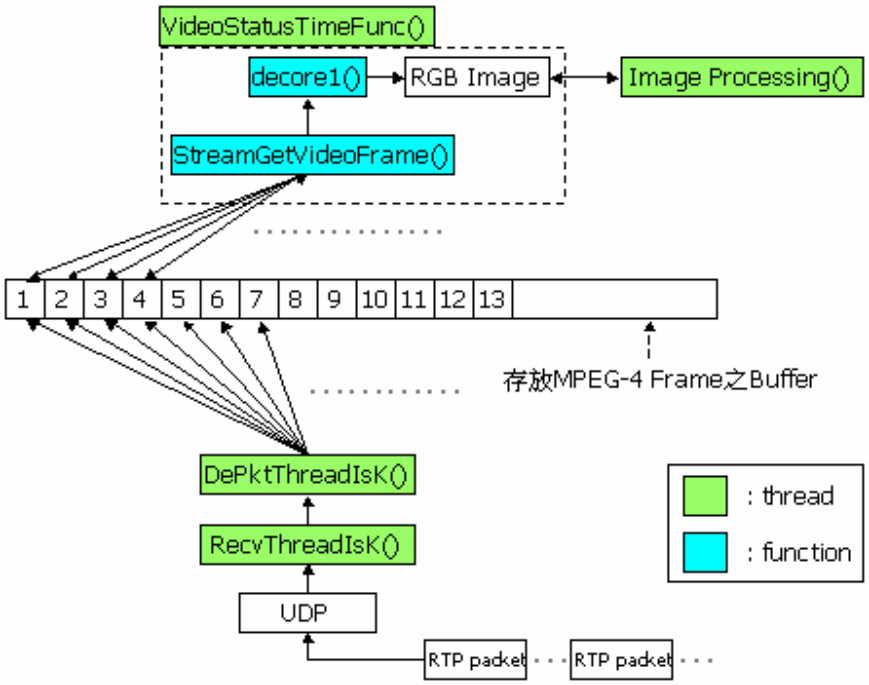


圖 6-5-1 Client 端之 Thread 關係圖

圖 6-5-1 為 Client 端之 Thread 關係圖，主要可分為四個 Thread 的架構，分別為 RecvThreadIsk()、DePktThreadIsk()、VideoStatusTimeFunc()、Image Processing()，各 Thread 的功能簡介如下：

RecvThreadIsK()：

主要工作為接收 Server 端傳送過來的 RTP packet。

DePktThreadIsK()：

主要工作為將 RTP packet 重組成 MPEG-4 Frame。

VideoStatusTimeFunc()：

主要工作為將儲存於 Buffer 內的 MPEG-4 Frame 取出來，並將它解碼成 Raw Data 以提供給影像處理程式處理或呈現在 Graphic User Interface(GUI)上。

Image Processing()：

影像處理程式。

各 Thread 的運作順序如下，首先 RecvThreadIsK()負責接收 Server 端傳送過來的 RTP 封包，進一步的利用 DePktThreadIsK()將 RTP 封包重組成 MPEG-4 Frame 並將此 MPEG-4 Frame 存放於 Buffer 內。VideoStatusTimeFunc()發現 Buffer 內有存放 MPEG-4 Frame 時，會利用 StreamGetVideoFrame()函式取出 MPEG-4 Frame，進而交由 MPEG-4 Decoder 解碼出 RGB Image，接下來 Image Processing()會針對此 RGB Image 進行影像處理的動作。

[Client 端系統穩定度之考量]

上述說明為 Client 端四個主要 Thread 的運作流程，然而為了使 Client 端程式能順利的執行，各 Thread 有其相關要求要達成，使 Client 端程式能穩定的執行。各 Thread 要求如下：

RecvThreadIsK()：

要能針對 Server 端傳送過來的 RTP 封包，快速且順利的接收。也就說此Thread接收RTP封包的速率要能配合Server端的資料傳輸速率。

DePktThreadIsK()：

當 RecvThreadIsk()每接收到一個組成 MPEG-4 Frame 所有的 RTP 封包時,要能迅速的將這些 RTP 封包重組成 MPEG-4 Frame。

VideoStatusTimeFunc()：

當 DePktThreadIsK()每組成一個 MPEG-4 Frame 時，要能迅速的將此存在 buffer 上的 MPEG-4 Frame 取出，並進行解碼的動作，以提供 RGB 影像給 Image Processing()使用。

會有上述針對 RecvThreadIsK()、DePktThreadIsK()、VideoStatusTimeFunc()的要求，主要是因為實際在執行 Client 端程式時，因為 Image Processing()內如果執行的是運算量很高的影像處理演算法時，此 Thread 會佔據很多 Operating System(OS)所提供的系統資源，進而連帶影響到其他 Thread 的運作。上述情形如發生的話，VideoStatusTimeFunc()會減少從 Buffer 取出 MPEG-4 Frame 及解碼的

次數，進而造成存放 MPEG-4 Frame 的 Buffer 越長越大，也就是說 VideoStatusTimeFunc()提供給影像處理程式的RGB影像並不是最新的。

[改善方法]

因為有上述問題存在，因此必須克服上述現象使 VideoStatusTimeFunc()有較多的機會去取出 Buffer 裡的 MPEG-4 Frame 並進行解碼的動作，使其能提供給影像處理程式最新的RGB影像。改善此問題的方法是設定各 Thread 的 Priority，使不同 Priority 的 Thread 運作頻率不同。方法如下：將需要花費大量運算時間的 Image Processing()設定為低優先權的 thread，需少量運算時間的 RecvThreadIsK()、DePktThreadIsK()、VideoStatusTimeFunc()設定為高優先權的 thread。Thread 的優先權設定完後，OS 會給高優先權的 thread 較高的執行頻率，低優先權的 thread 則給予較低的執行頻率。也就是說 RTP 封包的接收與 MPEG-4 Frame 的重組及解碼都可以配合傳送端的傳輸速率順利的執行並顯示在接收端，影像處理 Thread 則根據影像處理所需要的規格來抓取影像。

第七章 結論

在本論文裡，我們提出具有即時檔案傳輸及互動式性質的監控系統。在此系統架構下，監控人員並不需要直接在監控中心存取影像檔案，而是可藉由遠端即時傳輸的技術在遠端存取影像檔案。在人機互動性方面，因為我們有將物件定位技術、手勢辨識等影像處理加入監控系統中，使其提供相機的換手機制準則，及相關的事件定義。因此此監控系統可提供相機的自動換手機制，且當有事件發生時，錄影機制會自動開啓，使此平台達到人機互動性的要求，進而使監控工作更有效率的執行。

參考文獻

[1] REAL TIME PROTOCOL(RTP). Available:http://www.hellosoft.com/

[2] RTP Control protocol(RTCP).
Available:http://www.freesoft.org/

[3] REAL TIME STREAMING PROTOCOL(RTSP), mmsc, Draft RFC 2326
Available:http://www.ietf.org/

[4] Leggio Simone," Streaming Media over the Internet with Real Time Streaming Protocol", 2003.
Available:http://www.cs.helsinki.fi/u/jmanner/Courses/

[5] Cheng-Chang Lien, Sheng-Cheng Hsu," The Development of the Real-time Object Tracking Technology Based on the Image Mosaic of the Non-Stationary Scenes.

[6] Licsar, A.; Sziranyi, T.;" Dynamic training of hand gesture recognition system", Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on Volume 4, 23-26 Aug. 2004 Page(s):971 - 974 Vol.4

[7] Etienne Loupias and Nicu Sebe, "Wavelet-based Salient Points for Image Retriveval.

[8] C. Harris and M. Stephens," A Combined Corner and Edge Detector", Proc. of 4 th Alvey Vision Conference, 1988, pp. 147-151.

[9] Z. Zheng, H. Wang and E. Teoh," Analysis of Gray Level Corner Detection", Pattern Recognition
Letters, 1999, Vol. 20, pp. 149-162.

[10] M. Seki, T. Wada, H. Fujiwara, K. Sumi, "Background detection based on the cooccurrence of image variations",

Proc. of CVPR 2003, vol. 2, pp. 65-72.

[11] Gutchess, D.; Trajkovics, M.; Cohen-Solal, E.; Lyons, D.; Jain, A.K.; "A background model initialization algorithm for video surveillance",Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on Volume 1, 7-14 July 2001 Page(s):733 - 740 vol.1

[12] M. D. Huang, and L. H. Chen, "Two New Surveillance Systems," Proceedings of the 15th IPPR Conference on CVGIP, Taiwan, 2002.

[13] Rafael C.Gonzalez, Richard E. Woods,"Digital Image Processing",2002,Second Edition.