多重查詢方式之影片資料庫系統

陳珮宜　謝文雯　陳良弼
清華大學資訊工程研究所
{mr894308, mr894341, alpchen}@cs.nthu.edu.tw

摘要

近年來由於電腦軟硬體設備的進步與多媒體技術的蓬勃發展，使得多媒體應用成為當前非常熱門的領域，如何有效率的檢索多媒體資料是一個重要的研究課題。本論文主要是擴充我們之前的一個視訊檢索系統模組－3D-List，利用視訊資料的內涵建立索引結構來做查詢處理；採用藉由範例查詢（Query by Example）的方式，讓使用者直接在系統的介面上下查詢，不但可以表現各個物體分別所在的位置外，還提供時間的查詢。我們改進的部分首先將欲查詢物體的相對位置列入考慮，加入描述兩物體遠離（far away）及不相交（disjoin）的關係，除此之外，採用多重解析度（multi-resolution）的查詢處理，並修正索引表格（index table），使整個系統更能貼近使用者的需要。

關鍵詞：3D-List, spatial / directional relationships, multi-resolution, approximation

一、背景及目的：

視訊資料的內容包含了大量的資料，特別是包括了許多內涵物件之間時間以及空間上的關係，但是著重於這方面特質的研究並不多見。而近年來，在研究視訊檢索（Video Retrieval）的領域中，有許多技術越來越成熟，例如自動分割影片場景、自動辨識並擷取影片中的物件、物件軌跡分析等等，由於這些技術的發展，使得我們可以用更進階的角度去研究視訊文件的內容組成，從而發展出語意化查詢（Semantic Retrieval）的概念。Oomoto 以及 Takank 等人[5]曾經針對視訊資料內容中的資料模型以及儲存結構加以探討，並且提出一個視訊資料庫系統，稱之為 OVID (Object-oriented Video Information Database)。Liu 和 Chen[1][2]提出了一個多媒體系統的資料模型，整合了視訊物件之間在時間與空間上的關係，並利用了 3D-List 來加速查詢處理的進行。Lin 和 Chen[4]提出了一個視訊資料模型，定義了視訊物件的屬性，並且利用彈性的邊界配合狀態移轉圖 (State Transition Diagram) 從物件的移動軌跡推導出相對的移動特性。

本論文希望能加強 3D-List 的功能，考慮對視訊資料中物體相對位置的描述，並提出多重解析度的概念，讓使用者能夠使用範例查詢的方式，有效率的利用視訊資料。

二、簡介 3D-List

3D-List [2]是一個用來做視訊資料庫內查詢的資料結構。由於視訊資料較一般的媒體豐富許多，物體之間除了空間上的關係：X 軸及 Y 軸之外，還包含了時間先後的關係。因此在作視訊資料庫的查詢時，便可將查詢的條件轉為此三維：X、Y 及 T 的資訊，並利用 3D-List 這個資料結構來加速查詢的結果。

3D-List 共定義四種物體間的關係以及三種查詢形式以方便查詢。以下簡單列出四種關係：

1. adjacent relationship：兩個在 X、Y、T 軸中某一軸的座標值的距離為 n，可表示成 $I_1|_n I_2$ 。

2. appositional relationship：兩個在 X、Y、T 軸中某一軸的座標值相同的物體 I_1 及 I_2，可表示成 $I_1 \equiv I_2$。
3. Precedent Relationship：兩個物體 I_1 及 I_2，在 X、Y、T 軸中某一軸的座標值 $I_1 < I_2$，可表示成 $I_1 \Rightarrow I_2$。
4. unknown Relationship：兩個物體 I_1 及 I_2，在某軸的關係為未知，則表示為 $I_1 ? I_2$。

查詢型式分以下三種：

1. Q-type=0：所有的關係皆為 unknown。
2. Q-type=1：將 adjacency 轉成 precedent。
3. Q-type=2：保留 adjacency 及 apposition 關係。

根據不同的查詢型態，將查詢轉成分別在 X、Y 及 T 軸的關係，再使用 3D-List 找出所符合條件的資訊。

3D-List 的優點主要有下列兩點：

(1)可以同時查詢 X、Y 及 T 的關係；

(2)會建立 equivalent group：在建立 3D-List 的時候，為了避免搜尋整個資料庫，會建立" equivalent group," 如此可省下搜尋的時間。這也是 3D-List 一項較有效率的優點。

三、3D-List 在相對位置/方向上的延伸

相對位置及方向的查詢可能在某些方面較能接近使用者的需要[3]。若使用者比較重視兩個物體的相對位置，而不是兩個物體分別所在的位置的話，那使用原來的 3D-List 沒有辦法達到這個目的。

3D-List 裡其實已經有 position/direction 的概念，只是因為在處理時，spatial 的資訊被分散在 X、Y 裡面，所以較看不出來兩個物體間的距離遠近或是角度關係，如 Fig 1 所示，我們可以在查詢介面裡表示 Tree 跟 Horse 是屬於 DJ(disjoin)，Eagle 跟 Tree 是 FA(far away) 的關係，以及 Horse 跟 Eagle 的夾角等等，但是分別就 x 及 y 軸來看就並不是那麼明確，像 Tree 跟 Eagle 的關係必須分別就 x 及 y 去檢查是否合乎 precedent 的關係，而 precedent 也還需要加上判斷的門檻值，有許多部分是原來 3D-List 沒有定義的。

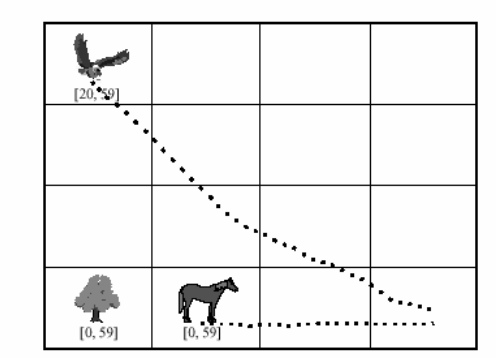


Fig. 1 物體遠近及相對位置的例子

為不破壞原有的 3D-List 架構，我們直接新定義查詢型式，並新增 operator 以方便使用者查詢。

原先的 3D-List 裡有三種查詢型式：

- Q-type 0 : unknown relationship
- Q-type 1 : adjacency→precedent
- Q-type 2 : adjacency and apposition relationship are retained.

在此針對空間上的關係定義新的 Q-type，將其定名為 Q-type 3 及 Q-type 4：

- Q-type 3 : **FA** and **DJ** relationship
- Q-type4 : query by angle

Far away 及 disjoin 可套用在空間關係及時間關係，但是因為主要是希望能增強在空間查詢的效能，為避免混淆，此處 FA 及 DJ 皆指的是空間關係。

(1) Q-type 3 : FA and DJ relationships

Definition 1：$\overset{FA}{\rightarrow}$

兩個物體 I_1 及 I_2，在 X、Y 軸中某一軸的關係為 far away，可表示成 $I_1 \xrightarrow{FA} I_2$。

Definition 2：$\overset{DJ}{\rightarrow}$

兩個物體I_1及I_2，在 X、Y 軸中某一軸的關係為 disjoin，可表示成$I_1 \overset{DJ}{\longrightarrow} I_2$。

Definition 3：FA threshold

在定義"Far away"關係時，需要定義門檻值，超過此門檻值的兩物體則

Definition 4：DJ threshold

Disjoin 的門檻值定義在大於 0 及小於 FA threshold。

Definition 5：Equivalent group of $A \overset{FA}{\rightarrow} B$，其中 A 及 B 分別是兩種 icon。

對屬於 A 這個 icon 的每一個 symbol 值，我們進行以下的動作：

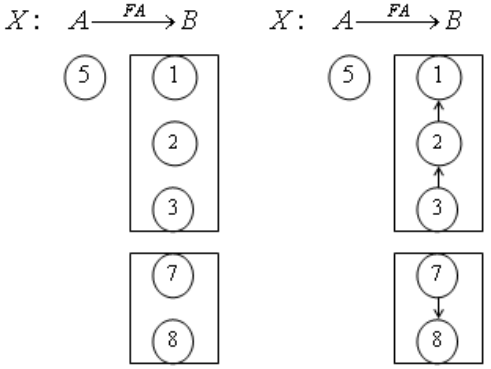
(i) 用 A 的 symbol 值，將 B 的 symbol 值分成比 A 小及比 A 大兩群，如 Fig. 2 所示。

(ii) 兩群中，比 A 小的那群，equivalent group 的建法是由大連到小，比 A 大的那群由小連到大，形成兩群。如圖 Fig. 3 所示。

(iii) 根據 FA threshold，在小的那群中根據群的方向由大到小檢查，一但發現 A 與 B 的差值大於 FA threshold 時便加上連結，並停止往下檢查。

(iv) 同樣的，在大的那群中根據群的方向由小到大檢查，一但發現 A 與 B 的差值大於 FA threshold 時便加上連結，並停止往下檢查。如 Fig. 4 中所示，此例的 FA threshold 為 3。

(v) 對每個 A 的值都需要以上述步驟建立 equivalent group，如此即完成。



左：Fig. 2 Equivalent group of FA 建法 Step 1

右：Fig. 3 Equivalent group of FA 建法 Step 2

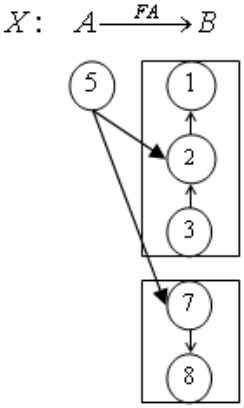


Fig. 4　Equivalent group of FA 建法 Step 4

FA operator 的討論：

1.　若 A 的 symbol 值都很接近（表示基本上 A 的變化不大），而 B 的 symbol 的變動很大，則可將 A 的 symbol 都群集在一起，用 A 這群的平均值來作 B 的分群。

此種類型可以想成一動一靜的兩個物體，使用者下查詢要查何時兩者的關係夠遠的情形。

A 的 symbol 與 B 的 symbol 值接近者也可以考慮刪掉，可以減少計算，但是要考慮 A group 內的差距，否則會有誤差甚至有 false dismissal 的情況發生。

若 A 的 symbol 值變化很大，則還是需要每個值去做分群動作。

2.　如何才能增進 filter 的效果？

分 equivalent group 的原因就是希望能夠減少不必要的計算，增進 filter 的效果。但是以上述的方法，【分大小兩群→分別作 group 的連結→檢查兩群中分別滿足 FA threshold 的值，並加上連結】勢必比起之前 3d-list 的 filter 的效果來的差，最差的狀況就是所有兩兩 A 及 B 的值需要兩兩組合的去做檢查。因此這裡提出可以改進的方法：

在分群時，如 Fig. 2 所示，可以分別紀錄其最大值及最小值，在 Fig. 4 要做連結時，在較小的那群可以檢查最小值，若最小值跟 A 值的距離比 FA threshold 來的小，表示這整個群和 A 值都不會有 far away 的關係，則不需要檢查這群。較大的那群可以檢查最大值，若最大值跟 A 值的距離比 FA threshold 來的小，則不需要檢查這群。

Definition 6：Result set of $A \overset{FA}{\rightarrow} B$

在我們分別對兩個軸做完 FA operation 後，由於只要其中一個軸的關係為 far away，整體的關係即為 far away，故將兩個軸的結果合併即為答案。

Definition 7 : Equivalent group of $A \overset{DJ}{\rightarrow} B$，其中 A 及 B 分別是兩種 icon。

大致上與 FA 的 equivalent group 的想法相同，詳列如下：

對屬於 A 這個 icon 的每一個 symbol 值，我們進行以下的動作`(此處 DJ threshold=3)

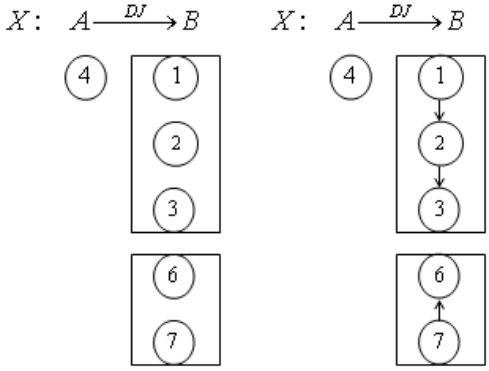
(i) 用 A 的值，將 B 的值分成比 A 小及比 A 大兩群，如 Fig. 5 所示。

(ii) 兩群中，比 A 小的那群，equivalent group 的建法是由小連到大，比 A 大的那群由大連到小，形成兩群。如圖 Fig. 6 所示。

(iii) 根據 FA threshold，在小的那群中根據群的方向由小到大檢查，一但發現 A 與 B 的差值在於 DJ threshold 之內時便加上連結，並停止往下檢查。

(iv) 同樣的，在大的那群中根據 group 的方向由大到小檢查，一但發現 A 與 B 的差值在於 DJ threshold 之內時便加上連結，並停止往下檢查。如 Fig. 7 中所示。

(v) 對每個 A 的值都需要以上述步驟建立 equivalent group，如此即完成。



左：Fig. 5 Equivalent group of DJ 建法 Step 1

右：Fig. 6 Equivalent group of DJ 建法 Step 2

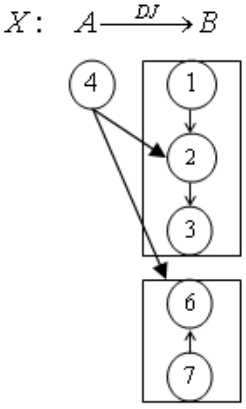


Fig. 7　Equivalent group of DJ 建法 Step 4

Definition 8：Result set of $A \overset{DJ}{\rightarrow} B$

Disjoin 的關係，必須要在 X 和 Y 軸都成立的情況下，兩個物體才能算是 disjoin。所以最後的 result set 必須將 X 軸的結果與 Y 軸的結果作交集，才是最後的結果。

DJ operator 的討論：

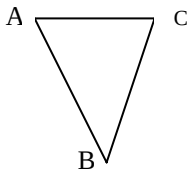
我們所定的 DJ threshold 指的是兩個物體間的歐幾里得距離（Euclidean distance），但是在 3D-List 裡，是分 X 軸及 Y 軸來檢查的，如此兩個物體可能實際上不屬於 DJ，但是分在 X 及 Y 裡就都是 DJ 了，ex: DJ threshold=4，x=4, y=4 的物體會被判斷為 disjoin，但是實際的距離已超過 DJ threshold→false alarm 發生，因此若要精準的答案便需要在做檢查，但是此方法可以確保沒有 false dismissal 的情況。

關於 FA and DJ operator 的其他討論：1.

$$A \xrightarrow{FA} B \xrightarrow{FA} C \equiv \left(A \xrightarrow{FA} B \right) \cap \left(B \xrightarrow{FA} C \right) ?$$

A 跟 B 是 FA 關係，B 跟 C 是 FA 關係，但是 A 跟 C 不見得就是 FA 關係，FA 並不能滿足遞移律。

EX:



A 跟 B 在 y axis 屬於 FA 關係，C 跟 B 在 y axis 也屬於 FA 關係，但是 A 跟 C 在 y axis 是相等的，在 x axis 是 DJ 的關係，故不存在遞移律

2. 找到更好的 equivalent set 的建法：
為了能延續 3D-List 的優點，我們希望能在 equivalent group 方面多做發揮，希望能有更好的 filter 的效果。

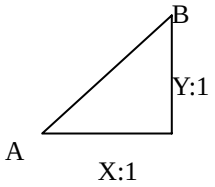
3. 如何做 FA 及 DJ 的近似？基本上 FA 及 DJ 便是一種比較類似 approximation 的概念，他把實際距離用相對的關係來取代，所以已經有了 approximate 的想法。同時我們也可以在 FA 及 DJ threshold 限制下，在放寬其值來作 approximation，應該也會有其效果。

(2) Q-type 4 : query by angle

希望能在知道兩個物體的夾角，以及兩者之間的距離時，能夠在資料庫中快速找到符合這樣條件的資訊。

先使用一個例子來說明想法：

$$A \xrightarrow{\theta=45^\circ, D=\sqrt{2}} B$$



θ代表的是 A 跟 B 之間的夾角，D 是兩者之間的距離，拆解成 X 及 Y 軸資訊後如下：

$$X : A \xrightarrow{D=1} B$$

$$Y : A \xrightarrow{D=1} B$$

這種情況等於是必須要將所有距離為 1 的所有組合找出來在做交集，而且角度資訊會被拆解成 x 及 y 的資訊了。在沒能有其他更好的方法來作 equivalent group 的情況下，我們覺得 3D-List 並不適合做 query-by-angle。使用其他的索引結構可能會得到比較好的效果。

四、多重解析度（multi-resolution）概念

我們的系統是採用藉由範例查詢的方式，讓使用者直接在系統的介面上下查詢，除了可以表現出各個物體的相關位置外，還提供時間及軌跡的查詢；這樣子的查詢方式會比文字敘述更貼近使用者想要得到的結果，只是根據圖示的大小、擺飾位置等等的不同，造成誤解的機率也相對的提高。在原先的論文[2] 中是將介面固定分成 4x4 的格子狀，如 Fig.8 所示，這樣子雖然可以約略的將物體做一些有效的運算（appositional relationship “≡” 和 precedence relationship “⇨”），使查詢處理更快速簡便；但是有時候 4x4 的分隔方式對整張

圖片而言太過於粗糙，不能夠適用於每一種查詢，因此我們想到用多重解析度的方式來改善這個缺點。

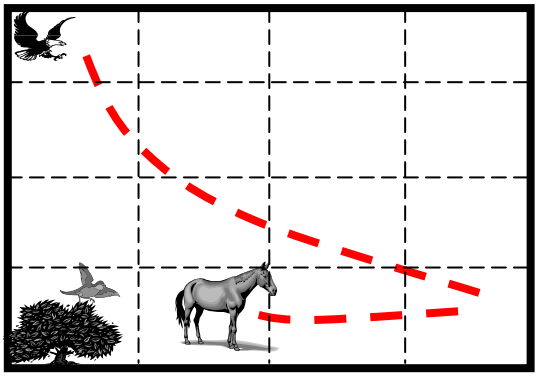


Fig. 8 使用者的查詢套用在 4x4 解析度

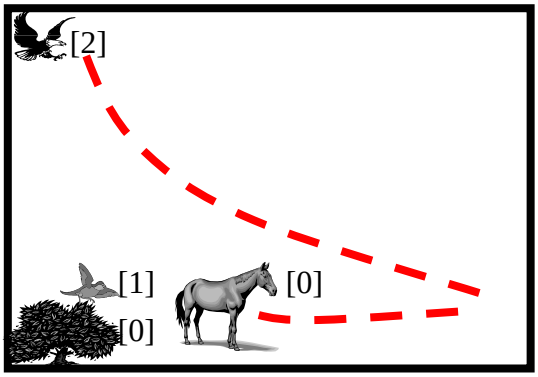


Fig. 9 使用者下查詢的範例

首先在介面處理方面，我們將原先 4x4 的分隔去掉，使整個介面在一開始的時候呈現空白的狀態，讓使用者自由放置物體圖示，而且圖示的大小也可以讓使用者自行調整到最合適的狀態。Fig.9 是一個查詢的例子，其中距離圖示最近的數字表示該物體出現的順序，由 0 開始編號，使用者還可以畫出物體的運動方向來做軌跡的查詢，另外可以特別指定哪兩個物體的關係為 far-away 或是 disjoin 等等關係。

由於3-D List最主要的想法在於將複雜的計算簡單化，利用物體之間的相對位置做查詢；我們發現在原本4x4的查詢介面上僅能區分出兩種相對位置關係：appositional relationship和precedence relationship，其他較為精細的關係卻被忽略掉了，雖然原本粗糙的查詢方式可以確定不會發生 false dismissal的情形，但是過多false alarm的查詢結果卻會使系統的精確度降低。在我們對系統的改進方法中，主要是探討precedent relationship的細分，因為appositional relationship的細分方式都必須去計算兩個圖示中心點的實際距離加上個別MBR(Minimum Bounding Rectangle)的重疊部分做判斷，這樣子一來和3-D List的基本想法就有違背，而且由於使用者畫出來的圖形大部分都只是個「約略」的雛形，套上過於精細的計算就很容易發生嚴重的false dismissal。

Table 1 索引表格中的資訊以及在不同解析度時的座標值

Video_oid	Icon_oid	Symbol_oid	Frame	X	Y	4×4		8×8		16×16	
						X	Y	X	Y	X	Y
V0001	E	S0001	54	26	423	0	3	0	6	0	13
V0001	E	S0002	45	27	107	2	0	4	1	8	3
V0001	E	S0003	20	141	155	1	1	2	2	4	4
V0001	B	S0004	25	55	219	0	1	0	3	1	6
V0001	B	S0005	31	78	366	0	2	1	5	2	11
V0001	T	S0006	1	24	267	0	2	0	5	0	8
V0001	T	S0007	25	100	230	0	1	1	3	3	7
V0001	T	S0008	30	349	88	2	0	5	1	10	2
V0001	H	S0009	40	301	212	2	0	4	3	9	6

V0001	H	S0010	30	172	147	1	1	2	2	5	4
V0001	H	S0011	22	364	169	3	2	5	4	11	8
V0001	H	S0012	1	101	355	0	2	1	5	3	11

主要是在索引表格（index table）的部分做修正，使得查詢處理可以選擇不同的模式。原來的索引表格包括 Video_oid、Icon_oid、Symbol_oid、X、Y 以及 Frame 的資訊，其中 Video_oid、Icon_oid、Sybmol_id 分別表示 Video 的 ID、Icon 的 ID，以及 Symbol 的 ID；X 及 Y 是表示在 4x4 分隔中落在那一格，其中 0≦X≦4，0≦Y≦4；Frame 則記錄該筆資料的 frame number。在新的 Table 中，我們會記錄 symbol 的中心點實際座標，另外根據不同的解析度計算該中心點座落於哪一座標單位格當中，如 Table 1 所示。

我們假設可以由各種已知的方式得到 Video_oid、Icon_oid、Symbol_oid、Frame、X 和 Y 的資訊，這裡的 X 和 Y 分別表示物體圖示中心點的 X 座標以及 Y 座標的實際值，利用 X 和 Y 可以計算出剩下的資訊，即物體的中心點在 4x4、8x8、16x16 不同分隔中落在哪一格。由於一開始的查詢介面沒有分格線，圖示的大小也沒有固定，因此使用者所下的查詢經過上述的計算會發生一個圖示橫跨多個格子的情形，為了使之後查詢處理方便起見，只紀錄中心點所坐落的格子點，當圖示的中心點恰好落在分隔線上時，我們採用隨機的方式決定它該屬於哪一個格子，為了防止 false dismissal 的情形出現，我們提出 Approximation 的方式，將在下面做詳細的介紹。

至於我們採用這三種解析度的原因，是用二的次方（power of two）的切割方式對一張圖片來說最為公平；以 $2^1 \times 2^1$ 來看，恰好將整張圖形分別在 X 軸方向以及 Y 軸方向對半切割，應該可以視為最粗糙的切割方式，但是因為在這個解析度下做查詢處理出來的效果可想而知不會太好，加上相同的結果也可以利用 $2^2 \times 2^2$ 這個解析度做 approximation 之後得到，因此我們直接用 $2^2 \times 2^2$ 做為第一種解析度；最精細的解析度採用 $2^4 \times 2^4$ 不再分割下去的原因則是我們覺得太多層的解析度會使計算量增加，而且繼續切割到太細的格子很容易使同一個圖示橫跨太多個格子，這樣反而會使精確度降低。

(1) Approximation

Approximation 的想法除了彌補只紀錄中心點位置的不足之外，還考慮到使用者所下的查詢並不是百分之百精確，因此我們想到利用 approximation 的方式達到更好的查詢處理。

由 3-D List 的原始觀念繼續延伸，我們將圖示中心點四周圍的格子都列入考慮，也就是 X 座標格或 Y 座標格和中心點只相差 1 的情形。以 Fig.10 為例，假設 Horse 的中心點位置落在(3, 1)當中，做 approximation 時就會將中心點落在(0,2)、(0,3)、(0,4)、(2,1)、(4,1)、(2,2)、(3,2)、(4,2)的 Horse 都當作是查詢處理之後的可以成立的答案，這樣子可以避免掉 false dismissal 的情形。當然，使用者下查詢時，可以選擇是否要採用 approximation 的方式。

接下來討論的是 T 軸上的 approximation，在原來的 paper 當中是希望使用者下查詢的時候可以直接指出物體出現的 frame number，但是可以發現這樣子的查詢方式並不合理，使用者光從影片無法獲得這個資訊，因此我們讓使用者下時間查詢時僅標明物體出現的先後次序，由 0 開始，數字越大表示物體出現的時間越晚。

(2) Query Processing

介紹改良系統的查詢處理方式之前，我們先定義 far-away 在查詢處理的定義：由於可以讓使用者指定哪兩個物體為 far-away 的情

形，加上我們有三種不同的解析度，為了簡化計算，我們分別定義了不同的 threshold，在 4×4 分隔中，單一方向要相差 2 格以上的距離，在 8x8 分隔中，單一方向要相差 4 格以上的距離，在 16x16 分隔中，單一方向要相差 8 格以上的距離，也就是要距離相差整張圖形的一半以上，至於詳細的演算法在 Section 3 就介紹過了。

接下來是查詢處理的步驟：

Step 1：由使用者下的查詢可以在 X 軸方向，Y 軸方向，以及 T 軸方向分別得到類似“A≡B⇨C”的表示方法，其中大寫的英文字母為 Icon_oid，兩個 Icon_oid 中間利用 appositional relationship 或 precedence relationship 做連結。先將三個軸分別做 Step1～Step7 的處理。

Step 2：利用索引表格內的資訊，分別建立這三個 List。首先確定使用者是否有指定要進行哪一種解析度的查詢，如果沒有，則從 4x4 開始，如果有，則直接由 Table 找合適的資料。然後將符合 Icon_oid 的特定軸資訊由 Table 當中挑選出來，用 node 的方式表示，裡面的值紀錄 Table 中適合的資訊，而 node 的下方都要註明該值屬於哪幾 Symbol，依值的大小由小而大排在 Icon_oid 的下面。

Step 3：先判斷垂直部分的 link，這部分的 link 都是由上到下的”↓”；如果 Icon_oid 左邊的 relationship 為“⇨”，則每兩個 node 中間都加入↓的 link；其他 Icon_oid 下面的 node 則是有相同值得時候才加入↓的 link，被↓串起來的 node 們形成一個 equivalence set。

Step 4：水平部分的 link 方面則都是由左到右的“→”，如果 Icon_oid 左邊的 relationship 為“⇨”，若左邊 Icon_oid 下面的 node 值小於這個 Icon_oid 下面的 node 值，由左到右在兩個 node 中間加入→的 link；如果兩兩 Icon_oid 中間的 relationship 為“≡”，則只有在兩邊 Icon_oid 下面的 node 值都相同的情況才可以加上→的 link。需要特別注意的是加 link 時，左邊的 node 如果有 equivalence set 的情形，水平的 link 要由該 set 最下面的 node 出發。

Step 5：檢查 far-away 的情形。兩 Icon_oid 之間的 relationship 若為 far-away，要移掉不符合的“→”。首先判斷 far-away 的兩個 Icon_oid 在判斷式中是否相鄰，如果相鄰，可以直接套用之前介紹的方法將不符合的“→”移除；如果沒有相鄰，則計算兩 Icon_oid 在判斷式中間隔了多少個“⇨”，每隔一個，threshold 的值就減 1，之後利用新的 threshold 套上之前介紹的方法判定 far-away 的情形。

Step 6：如果使用者不做 Approximation 的查詢處理，則直接跳到 Step 7；否則對 X 軸以及 Y 軸的 list 進行下面的處理：如果 Icon_oid 左邊 relationship 為“⇨”，則左邊 Icon_oid 下面 node 的值等於這個 Icon_oid 下面 node 的值時，在這兩個 node 當中加上→的 link；如果兩 Icon_oid relationship “≡”，則左邊 Icon_oid 下面 node 的值和右邊 Icon_oid 下面 node 的值相差 1 的情況都加入→的 link。

Step 7：移走沒有被 link 到的 node。

Step 8：利用 Symbol_oid 將三個軸做 join 的動作，留下來的 node 可由 T 軸的部分得到 video clip。

Step 9：如果使用者下的查詢並不包括軌跡的

部分，則直接跳到 Step10；否則進行下面的處理：以 T 軸為基準，檢查 Symbol 的 X 軸及 Y 軸座標是否符合該 Symbol 的查詢軌跡（例如 X 軸越來越大，Y 軸越來越小，或是 X 軸先變小再變大等等），如果不符合，將此段 clip 移除。

Step 10：Output 最後剩下的 clips，讓使用者判斷是否符合需求，如果不符合，可以讓使用者依結果調整不同的解析度，然後對 X 軸以及 Y 軸再重複 Step1～Step10，T 軸維持不變。

Step 11：Output 的結果若牽涉到 approximation 的處理時，必須去做 rank 的動作，我們計算 rank 的方式如下列公式(一)所示，Rank 越高表示越接近使用者下的查詢：

$$Rank = \frac{\text{不爲}approximation\text{的}link\text{個數}}{\text{整條路徑所用的}link\text{個數}} \times 100\%$$

(3) 圖示範例

下面所舉的例子是使用 Fig.9 為查詢、Table 1 為索引表格所得出的步驟，其中假設 E 和 T 之間存在著 far-away 的關係，列出 4x4 和 8x8 在三軸最終的 list，其中虛線的箭頭表示 approximation 的部分：

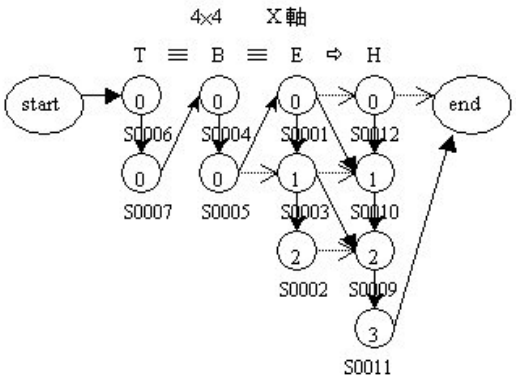


Fig. 10 4x4 解析度下的 x 軸 List 範例

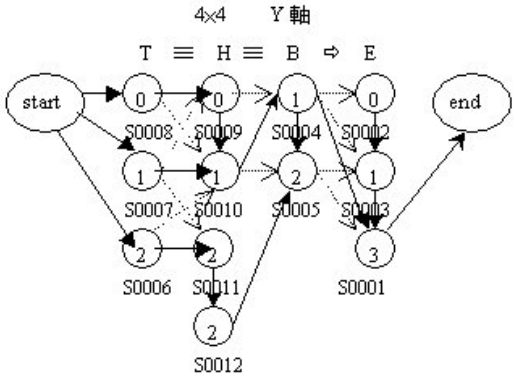


Fig. 11 4x4 解析度下的 Y 軸 List 範例

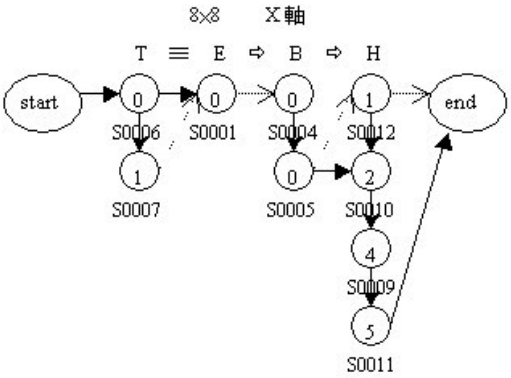


Fig. 12 8x8 解析度下的 x 軸 List 範例

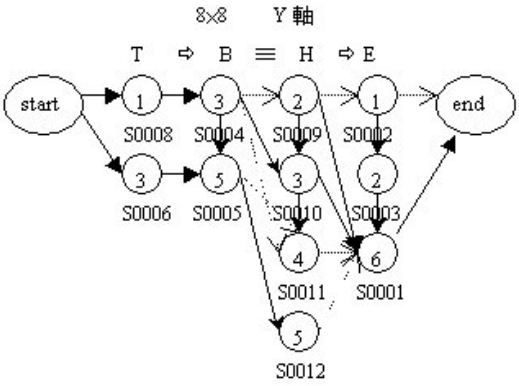


Fig. 13 8x8 解析度下的 Y 軸 List 範例

(4) 討論

由上面這個例子我們可以很明顯的看出來當解析度變高的時候，所得到的解也會變少，因為條件變的更嚴格，所以可以順利的過濾掉一些資料。而究竟在不同解析度中加入 approximation 的 link 會不會影響我們之前的

假設呢？答案是不會，我們來看下面的例子：

Table 2 不同解析度之不同座標值的例子

4x4 的 X 軸	8x8 的 X 軸
1	0
2	3

假設這兩個 node 都是“≡” relationship，Table 2 在 4x4 的 X 軸部分，這兩個 node 存在著 approximation link，因為彼此坐落在相差只有 1 的格子中；但是當相同的 node 以 8x8 解析度表示時，這兩個 node 有可能改變成坐落在相差大於 1 的格子中，此時兩個 node 並不存在 approximation 的 link。由此可知，在越高的解析度，link 各數越少，所得到的答案也較為精細，更接近使用者下的查詢。

以上面的例子在解析度 為 4x4 下做查詢處理，最後可以得到一組結果，{ S0001, S0005, S0006, S0012}，及 clip{Frame 1 ～ Frame 54}，但是相同的查詢在解析度 為 8x8 下卻找不到結果。

五、結論及未來工作：

首先我們加入了 Q-Type 3 和 Q-Type 4 的查詢，前者主要在定義 far-away 和 disjoin 的情情，後者主要在定義 query by angle 的情況；我們對這些方法提出了詳盡的演算法。接下來改進查詢介面，我們將原本的分隔線拿掉，讓使用者可以自行調整圖示的大小。另外加入多重解析度以及 Approximation 的觀念，讓查詢所得更接近使用者心中預設的想法；多重解析度可以讓使用者調整需要的精確度，Approximation 則是提供 rank 讓使用者找到相似的解；不但如此，還可以處理簡單的物體軌跡。目前為止我們並沒有考慮兩物體重疊之後的情形，例如兩物體呈現包含或是被包含的關係等等，如果使用者需要這些更精細的答案，依據現在的系統還無法做到，因為要紀錄每個物體的 MBR，每次依據使用者的要求再做更進一步的計算才有辦法得到這些要求，但是這樣一來就違背了 3-D List 原始的簡單計算原則，因此要如何做到這些查詢就需要更進一步的討論了。

六、參考文獻：

[1] C. C. Liu and Arbee L. P. Chen, “Vega: A Multimedia Database System Supporting Content-Based Retrieval,” *Journal of Information Science and Engineering*, Vol. 13, No. 3, Sep. 1997, pp. 369-398.

[2] C. C. Liu and A. L. P. Chen, “3D-List: A Data Structure for Efficient Video Query Processing,” to apper in *IEEE Trans. on Knowledge and Data Engineering*.

[3] Jae-Woo Chang and Yeon-Jung Kim, Spatial-match Iconic Image Retrieval with Ranking in Multimedia Databases"

[4] C. H. Lin and Arbee L. P. Chen, “Motion Event Derivation and Query Language for Video Databases,” in *Storage and Retrieval for Media Databases 2001, Proceedings of SPIE Vol. 4315*, pp.208-218.

[5] Eitetsu Oomoto and Katsumi Tanaka, “OVID: Design and Implementation of a Video-Object Database System,” *IEEE Transactions on Knowledge and Data Engineering*, August 1993, pp. 629-643.