

國立臺灣海洋大學

資訊工程學系

碩士學位論文

指導教授：張雅惠博士

航空旅客旅行路徑之空間分群研究

The Study on Spatial Clustering for  
Traveling Routes of Air Passengers

研究生：莊思彥 撰

中華民國 105 年 10 月

# The Study on Spatial Clustering for Traveling Routes of Air Passengers

研究生：莊思彥

Student : Si-Yen Zhuang

指導教授：張雅惠

Advisor : Ya-Hui Chang

國立臺灣海洋大學

資訊工程學系

碩士論文

A Thesis

Submitted to Department of Computer Science and Engineering

College of Electrical Engineering and Computer Scienece

National Taiwan Ocean University

In Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Computer Science and Engineering

October, 2016

Keelung, Taiwan, Republic of China

中華民國 105 年 10 月

## 摘要

隨著社會經濟水平和人們生活水平的提高，越來越多的人們選擇在節假日外出旅行，出國旅遊也成為近年來的熱門選擇。在這個背景下，再僅僅依靠旅遊業者為旅客設計旅行路徑已經不能滿足旅客的多樣化的需求。本論文將旅客旅行路徑資料作為分析研究對象，將複雜的路徑資料整合摘要化成有意義的旅客旅行路徑圖，如此可讓使用者得知其他旅客過去主要的移動路徑，讓使用者作為未來規劃旅遊路徑的參考。

本論文所遭遇到的主要困難點，在於路徑涵蓋整個地球且方向雜亂。我們所提出的方法，首先將原始旅行路徑資料與機場資料存入資料庫，以便 SQL 可以取出特定範圍的路徑。接著，將原始地圖切割成 cell，求出旅行路徑與 cell 的交點，並將交點座標依照航段方向彙整合併。在本論文中，我們總共使用 8 種方向去合併航段，但由於航段可能在 cell 中變化方向，所以我們同時記錄並保留中轉資訊。最後，由於平均後的線段，彼此不一定相連，為了呈現連續的移動路徑，我們參照原始旅遊路徑在 cell 中的移動方式並連接平均線段，得到一個新的旅客旅行路徑圖。

我們實作上述的方法，所建構的系統可以提供使用者以台北出發，到達美洲、亞洲、歐洲、非洲等的情況下，最多旅客的旅行移動方式。另外，我們並進行一系列的實驗分析所提方法的效率，由結果發現路徑數量大小、資料中航線所移動的範圍與 cell 數量都會影響系統耗時。

## Abstract

Traveling abroad has become a popular choice in recent years, but travel plans provided by travel agencies are unable to meet different needs. This thesis directly analyzes the traveling routes of air passengers, and summarizes them into meaningful routes. The user could therefore know the main traveling paths of previous travelers and use them as a reference for future traveling plan.

The main problem which needs to be solved is that the traveling routes cover the whole earth with different directions. The proposed approach is to first represent the original traveling routes and airport data into a database, so we can issue proper SQL queries to retrieve routes with specific directions and destinations. Then, we divide the whole world map into a 2D array of equal-sized cells, find intersection points of the traveling routes with cells, and merge the segments based on eight different directions. However, since flights may change directions within a cell, we keep transit information in the process of merging. Beside, the averaged flight segments of each cell are not always connected, so we also record the original travel route to determine how to connect them to present continuous paths.

We have implemented the proposed method, and the constructed system can provide users the previous major traveling routes from Taipei to America, Asia, Europe, Africa, respectively. We have also performed a series of experiments to study the efficiency issues. Experimental results show that the amount of traveling routes, the range covered by the routes and the number of cells all affect the performance of the system.

## 誌謝

首先，感謝指導教授張雅惠博士，對於本論文給予許多幫助，且在研究論文期間不時地共同討論，從老師的身上學到了精確嚴謹的做事態度，與解決諸多的問題，使學生能順利的完成論文。

除此之外，感謝臺北科技大學劉傳銘博士和林川傑博士百忙之中抽空參與論文審查工作，也感謝學弟們給予本論文意見與幫助，使論文更趨近完善。

最後，感謝韋錫跟承翰學長，帶領著我們適應實驗室環境與經驗的傳承。感謝我的同學蔚齊與學弟妹們，一起陪伴著我渡過研究所生涯歷練與成長。更要感謝我最愛的家人們，在學習階段給予無限的支持和精神上的鼓勵，讓我也能夠無憂的專注於研究，在此致上我最大的感激，謝謝你們。

# 目錄

<b>第 1 章 序論與技術背景 .....</b>	<b>1</b>
1.1 研究動機與目的.....	1
1.2 研究方法與貢獻.....	2
1.3 相關研究.....	3
1.4 論文架構.....	4
<b>第 2 章 資料前置處理階段 .....</b>	<b>5</b>
2.1 背景定義與問題說明.....	5
2.2 路徑資料介紹與範例.....	6
2.3 關聯式資料庫綱要.....	8
2.4 資料前置處理.....	12
<b>第 3 章 系統架構與資料前處理說明 .....</b>	<b>14</b>
3.1 系統架構.....	14
3.2 BulidCell 模組.....	15
3.3 BulidRoute 模組 .....	17
3.4 計算交點演算法.....	21
<b>第 4 章 航線整合與顯示 .....</b>	<b>24</b>
4.1 範例說明.....	24
4.2 GetMajorVector 模組介紹與 avgcell 說明 .....	25
4.3 link 和 turn 介紹與說明 .....	33
4.4 Merge 模組介紹與 resultroute 結構說明 .....	40
4.5 Display 模組與介紹 .....	46
<b>第 5 章 系統呈現與實驗結果 .....</b>	<b>48</b>
5.1 資料集說明.....	48
5.2 系統呈現說明與討論.....	49
5.3 cell 大小對輸出呈現的影響.....	55
5.4 cell 數量大小之實驗與效率評估.....	57
5.5 航線數量之效率評估.....	58
<b>第 6 章 結論與未來方向 .....</b>	<b>61</b>
<b>參考文獻 .....</b>	<b>62</b>

## 圖目錄

圖 1-1	交通部觀光局的觀光統計圖表 .....	1
圖 1-2	以桃園機場出發的旅客旅行路徑圖 .....	2
圖 2-1	復興航空航網圖 .....	6
圖 2-2	聯合國地理區 .....	9
圖 2-3	小分群特例 .....	10
圖 2-4	資料前置處理階段架構圖 .....	12
圖 2-5	資料前處理階段輸出範例 .....	13
圖 3-1	系統架構圖 .....	14
圖 3-2	cell 範例圖 .....	15
圖 3-3	BuildCell 模組 .....	16
圖 3-4	地球切割圖 .....	16
圖 3-5	BulidRoute 模組 .....	18
圖 3-6	調整航段的範例圖 .....	19
圖 3-7	adjust 演算法 .....	20
圖 3-8	求取兩直線交點的流程圖 .....	21
圖 3-9	intersect 演算法 .....	23
圖 4-1	未處理前的範例示意圖 .....	25
圖 4-2	航段方向種類範例 .....	26
圖 4-3	GetMajorVector 模組 .....	28
圖 4-4	完整交點座標範例 .....	31
圖 4-5	GetMajorVector 模組結果圖 .....	32
圖 4-6	MBR 相交檢查法 .....	32
圖 4-7	cell 間平均線段連接問題 .....	33
圖 4-8	移動方位 .....	34
圖 4-9	cell 內平均線段中轉問題 .....	35
圖 4-10	turn 轉折範例 .....	35
圖 4-11	航段經過 cell 順序範例圖 .....	36
圖 4-12	madeline 演算法 .....	38
圖 4-13	演算法範例 .....	41
圖 4-14	航段方向對應的 cell 方位 .....	41
圖 4-15	merge 模組 .....	44
圖 4-16	延長(extend)方法 .....	45
圖 4-17	Display 模組 .....	47
圖 5-1	2013 年台北至歐洲的原始旅客旅行路徑圖 .....	48
圖 5-2	台灣至歐洲的旅行路徑圖 .....	51
圖 5-3	台灣至亞洲與大洋洲的旅行路徑圖 .....	52

圖 5-4	台灣至美洲的旅行路徑圖 .....	53
圖 5-5	台灣至非洲的旅行路徑圖 .....	55
圖 5-6	cell 大小對系統呈現的影響實驗圖 .....	56
圖 5-7	cell 格數影響耗時之比較圖 .....	57
圖 5-8	不同格數大小的輸出矩陣比較圖 .....	58
圖 5-9	不同資料集之耗時比較圖 .....	59
圖 5-10	不同航線數量的耗時比較圖 .....	60

## 表 目 錄

表 2-1	資料說明與需求表 .....	6
表 2-2	資料範例說明表 .....	7
表 2-3	國家地區代碼範例 .....	7
表 2-4	機場 IATA 代碼範例表 .....	8
表 2-5	資料庫綱要 .....	8
表 2-6	分區對應表(國家為複數集合) .....	10
表 2-7	Query 範例 .....	12
表 3-1	cell 矩陣範例 .....	16
表 3-2	路徑資料檔(pathfile)範例 .....	17
表 3-3	route 矩陣範例 .....	17
表 3-4	adjust 範例 .....	20
表 4-1	完整範例資料 .....	24
表 4-2	avgcell 矩陣結構 .....	25
表 4-3	$d(\text{direction})$ 表示航段方向 .....	26
表 4-4	avgcell 範例 .....	29
表 4-5	檢查法比較表 .....	33
表 4-6	link 矩陣結構 .....	34
表 4-7	turn 矩陣結構 .....	35
表 4-8	排序說明 .....	36
表 4-9	swapcell 矩陣結構 .....	37
表 4-10	link 與 turn 範例結果 .....	39
表 4-11	resultroute 矩陣說明表 .....	40
表 4-12	resultroute 矩陣與 additional 矩陣內容 .....	45
表 4-13	javascript 語法說明 .....	46
表 5-1	不同資料集 .....	49
表 5-2	路徑筆數的變化 .....	49
表 5-3	不同 cell 大小之各模組耗時(ms) .....	57
表 5-4	歐洲資料集在不同格數所輸出之矩陣大小 .....	58
表 5-5	不同資料集各模組的耗時(ms) .....	59
表 5-6	不同航線數量下各模組的耗時 .....	59

# 第 1 章 序論與技術背景

在此章，我們先說明本論文的研究動機與目的，以及提出本論文的研究方法與貢獻，並介紹相關的研究，最後說明各章節的內容及本論文的架構。

## 1.1 研究動機與目的

近十年出入境台灣的人次逐年上升，依照中華民國交通部觀光局所公佈的觀光統計資料<sup>1</sup>，如下圖 1-1，可以得知台灣從 2005 年(民國 94 年)開始，出國的人次約有 820 萬多人，到 2014 年(民國 103 年)已成長至 1180 多萬人。然而，現今航空領域的論文大多分析航空公司或機場等，甚少針對航空旅客的移動路徑。



圖 1-1 交通部觀光局的觀光統計圖表

本論文將某航空運輸協會所提供的旅客旅行路徑資料作為分析研究對象，我們目前有 2012 年與 2013 年的資料，該資料包含起迄機場、航空公司、旅客數、機票價與收益等，詳細內容我們會在第二章說明。而下圖 1-2 為 2013 年從台北桃園機場出境的未處理的原始旅客旅行路徑圖，我們利用了該組織所提供的機場資料，繪製出了此圖。

<sup>1</sup> <http://admin.taiwan.net.tw/public/public.aspx?no=315>，中華民國交通部觀光局

我們發現該圖路線散亂，嚴重重疊，無法提供有意義的資訊。所以我們希望可以整理與彙整出從起點(如台灣桃園國際機場)、至迄點(如北京首都國際機場)且包含中轉機場(如香港國際機場)的路徑資料。我們藉由對路徑本身進行研究，將複雜的路徑資料，整合摘要化成有意義的旅客旅行路徑圖，如此可讓使用者從龐大的路徑資料中，得知其他旅客過去主要的移動路徑，讓使用者做為未來規劃旅遊路徑的參考。

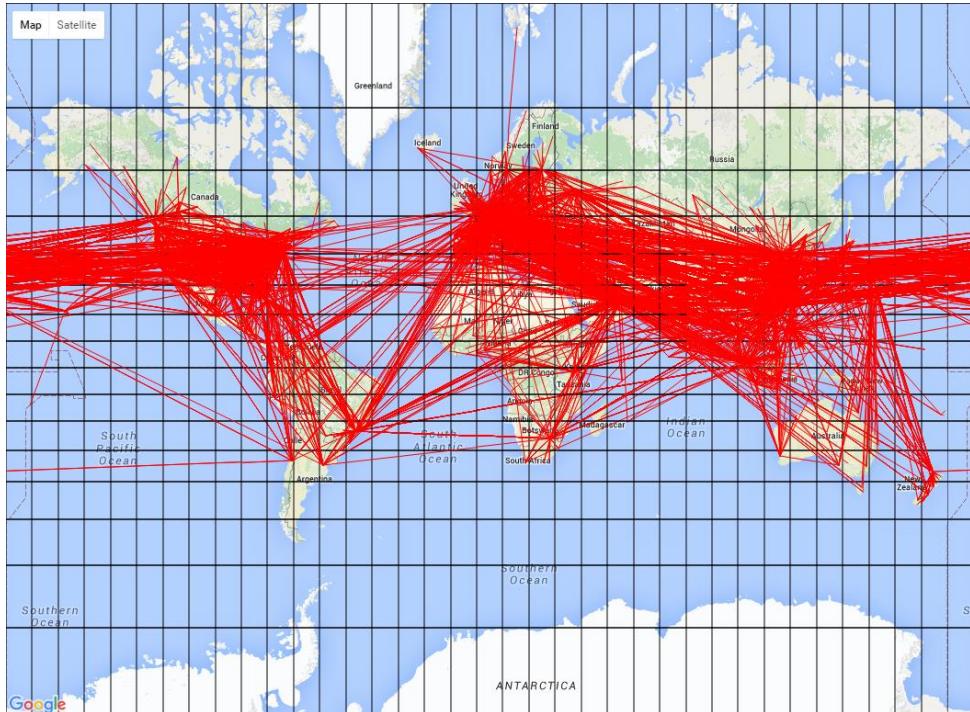


圖 1-2 以桃園機場出發的旅客旅行路徑圖

## 1.2 研究方法與貢獻

此研究的主要議題，在於如何整合複雜的路徑資料，並輸出摘要化的路徑，其困難點在於路徑涵蓋整個地球且方向雜亂。我們所提出的方法，首先將原始旅行路徑資料與機場資料存入資料庫，以便 SQL 可以取出特定範圍的路徑。接著，將原始地圖切割成 cell，求出旅行路徑與 cell 的交點，並將交點座標依照航段方向彙整合併。在本論文中，我們總共使用 8 種方向去合併航段，但由於航段可能在 cell 中變化方向，所以我們同時記錄並保留中轉資訊。最後，由於平均後的線段，彼此不一定相連，為了呈現連續的移動路徑，我們參照原始旅遊路徑在 cell 中的移動方式來連接平均線段，得到一個新的旅客旅行路徑圖。

由本系統所分析的結果得知，由台灣往歐洲的路徑，大部分的航線在亞洲的中轉方式主要有三種方向，分別往中國、東南亞與日韓，接著從亞洲移動到歐洲

後，以德法交界處為移動到其他歐洲國家的中轉區。由台灣往亞洲與大洋洲的路徑中，以移動到中國、東南亞與日韓為大宗。由台灣往美洲的路徑，大部分的航線都經過太平洋後抵達美國，其中的移動方式有兩種，分別為以日本中轉或直達美國。由台灣往非洲的路徑，以直接移動到西亞後轉南歐、與移動到東非後轉南非，這兩種為最多航線的移動方式。

### 1.3 相關研究

首先，我們探討航空與運輸領域相關的研究。在[陳13]的研究中發現，當不同航空公司提供旅客同一種航線時，旅客大多數會選擇形象較好的航空公司。[孫10]比較低成本航空公司與傳統航空公司，發現票價、行李限重、抵達時段以及起降機場的條件越好，低成本航空公司的票價就可以越接近傳統航空公司。[李13]則分析機場所在城市與旅客選擇的相關性，研究發現旅客大多數會選擇居住地所在的機場，作為啟程機場，而目的地機場則會選擇位於交通便利的城市。

除了上述研究旅客的選擇方向外，[呂09]將機場視為節點，把航線構成網路，最後利用不同網絡模組比較群聚度、路徑長度與節點連接數的重要性，並給出每個城市在航線中的發展潛力。[JQS14] 與[WZ15]也分別以網絡模組分析美國與中國機場與城市的關係，且城市本身的價值是否影響航線構成的網絡模組，在研究中發現大城市，其網絡模式多為輻射型，而小城市如果位於交通樞紐，其網絡模式為區域代理型(進入航線單一，離開航線眾多)。

其次，我們探討空間資料分群的研究。關於點的研究，[SMS11]提出了當分群過多時，如何減少可視化時地圖產生的雜亂。[林姚15]使用了以密度為基礎的DBScan 演算法將落雷發生地點分群。[GZJG12]則探討不同時間上，起迄點移動的變化性與所在地點的相互關係，該研究使用計程車所發送的GPS座標，將其建立成移動軌跡圖，然後分析在不同時段裡，點座標的密集度與移動的模式，最後得到不同地區在各種時段裡的重要性，例如火車站在上班時間迄點會十分密集，而在下班時間起點會十分密集，但在休息時段兩點都極為稀疏。

然後關於軌跡(trajectory)資料的研究有[MSZ13]，該研究藉由分區方法，將同區域重複的移動軌跡資料減少，以提升系統效率。[WLLP14]則分析船舶的移動路徑，提出將地區分格並產生主要路徑的方法。本論文也參考了該論文的方法，但不同之處在於該研究的移動路徑是記錄船舶所在座標與時間值，並將相同時間內的點座標彙整，將彙整後的座標構成一組主要移動路徑，但本論文的研究資料則不具有時間屬性。最後，[OSK14]則是研究在道路中，找出涵蓋最多活動的前  $k$  條最短路徑。該論文擴充  $k$ -means 的方法用於路網中，值得我們參考。

## 1.4 論文架構

本論文其餘各章節的架構如下：第二章先介紹本論文所研究資料背景與名詞定義，然後說明資料庫表格定義與所使用的 SQL 查詢句，藉以對本論文的研究有基礎的認識。第三章敘述相關的系統架構，以及模組如何讀取航線資料與切割 cell 範圍。第四章介紹摘要化路徑的方法，說明如何從相交的航線與 cell 中計算平均線段，然後將線段相連後使用 Google Map 呈現。第五章中討論本系統所呈現的摘要化路徑，並以實驗比較在不同大小的資料集與 cell 下的系統效率。最後於第六章提出本論文的結論與未來方向。

## 第 2 章 資料前置處理階段

我們在此章說明空運學的基本定義和本論文欲解決的問題，並接續介紹旅遊旅行路徑資料，包括各欄位的基本概念與資料格式，以及如何進行資料前置處理。

### 2.1 背景定義與問題說明

首先，我們介紹空運學中的基本名詞與背景，所謂空運(air transportation)乃是航空運輸的簡稱，也就是以航空器(air craft)為工具達到人或物之場所移動的目的。空運市場則是指空運企業(例如航空公司或物流公司)提供給顧客的運輸體系，比方說，桃園機場-香港機場、台北-高雄或日本-美國[謝 06]。接著，我們介紹與本研究相關的基本定義。定義如下：

**[定義 2-1]** 航段(Segment)：班機(flight)之起點與落點只有一次的行程[Pu06]。本論文將其定義為兩個機場間的“直線”移動路徑，並非為飛機實際飛行路徑。

**[定義 2-2]** 航線(Route)：指交通工具在兩個地點之間的固定移動路線，其中空中交通工具的移動路線，簡稱為飛行航線或航線<sup>2</sup>。由起迄機場與中轉機場構成，可包含複數個航段，且最多四個轉機機場、五個航段。

**[定義 2-3]** 旅行路徑(Path)：本論文所使用的旅客旅行移動之資料的名稱，簡稱為路徑，可視為航線附帶一些統計資料的旅客數，不過在本論文中“航線”與“路徑”可以混用。

一般航空公司會針對其提供的航線繪製航線網，或稱為航線網路圖，圖中的點代表機場或是城市位置，線則是代表所相連的兩點有運輸途徑，如圖 2-1。但是本論文的目的，是希望整合旅客搭乘各家航空公司的旅行路徑，如以下定義所示：

**[定義 2-4]** 問題定義：給定一組旅行路徑集合，本論文希望將其依照經過機場的地理位置，將其整合摘要化，以清楚顯示旅客喜好的旅行路徑。

---

<sup>2</sup> <https://zh.wikipedia.org/wiki/%E8%88%AA%E7%BA%BF>，航線

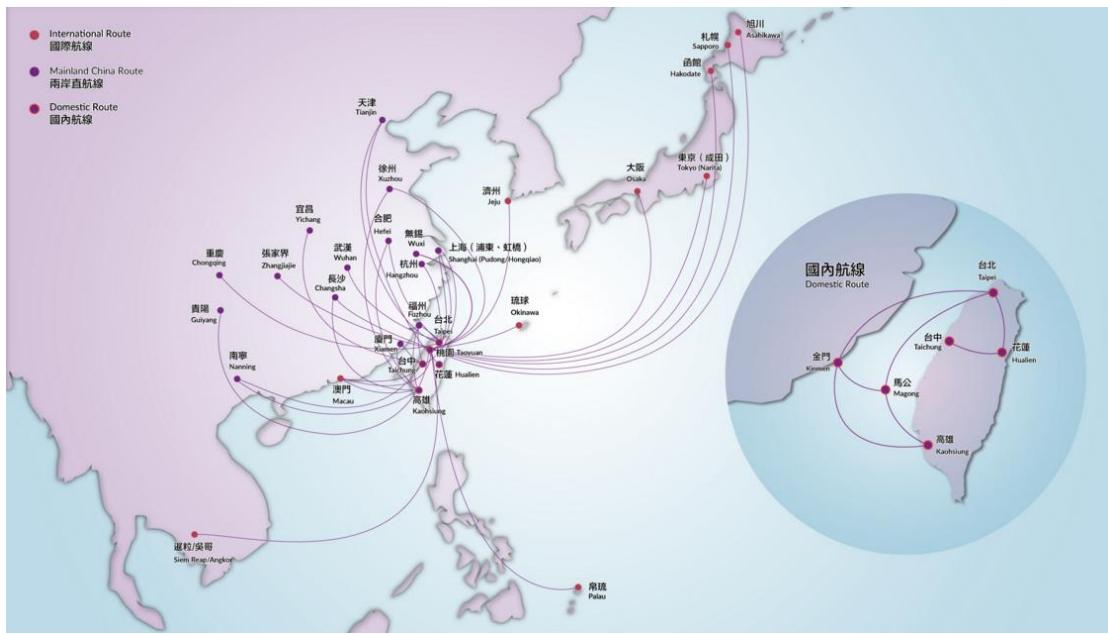


圖 2-1 復興航空航網圖<sup>3</sup>

## 2.2 路徑資料介紹與範例

本論文所處理的路徑資料來自某個國際性的民航組織，而以下資料則是該組織對於各航空公司旅客載量與機場銜接資料檢索之報告，其檔案格式為 Excel。基本格式包含航空公司、起迄機場、主要承運的航空公司、真實旅客數、預計旅客數、旅客數佔市場人數比例、平均票價與收益，如表 2-1 所示。

表 2-1 資料說明與需求表

欄位名稱	欄位意義	本論文是否需要使用
Dom AI	主要承運此航線的航空公司	否
AI 1	第一航線載送之航空公司	否
AI 2	第二航線載送之航空公司	否
AI 3	第三航線載送之航空公司	否
AI 4	第四航線載送之航空公司	否
AI 5	第五航線載送之航空公司	否，雖有欄位，但無資料
Orig	啟程機場	是
Stop #1	第一轉機機場	是
Stop #2	第二轉機機場	是
Stop #3	第三轉機機場	是

<sup>3</sup> <http://www.tna.com.tw/tw/information/flight-information/route-map>，復興航空航網圖

Stop #4	第四轉機機場	否，雖有欄位，但全無資料
Dest	目的機場	是
Reported Pax	真實旅客數	是
Reported + Est. Pax	預測旅客數	否
Pax Share	旅客數佔市場人數比例	否
Fare	平均票價	否
Revenue	收益=平均票價×預計人數	否

注意到，本論文只使用表 2-1 中的 6 個欄位，且我們研究的旅客旅行路徑資料，並非全球全部的路徑資料，而是以台灣為進出之旅客旅行路徑，一筆範例資料如表 2-2 所示。欄位中若無資料，則內容會以雙橫槓(--)表示，但起迄機場與旅客數不會無資料，旅客數可以為零。

表 2-2 資料範例說明表

欄位名稱	舉例	格式說明
Orig	TPE: Taipei, TW	機場代碼: 機場名,國家代碼
Stop #1	HKG: Hong Kong, HK	機場代碼: 機場名,國家代碼
Stop #2	FRA: Frankfurt, DE	機場代碼: 機場名,國家代碼
Stop #3	--	無資料
Dest	BUD: Budapest, HU	機場代碼: 機場名,國家代碼
Reported Pax	66	實際旅客數，為整數

檢索到的資料，機場會以 IATA 之標準 3 碼代號再加冒號，引出機場之全名(也就是城市名)，最後加上國家別之雙碼代號。此兩種代碼詳細說明如下：

- 國家地區代碼或稱國家地區編碼，簡稱國碼，是用來標誌國家的一組縮寫或符號，在此介紹的為 IATA 所使用的 ISO 3166-1，用於標記機場所在國家，但在此處也有標示非國家的地區代碼，如香港與澳門，如下表 2-3 所示。

表 2-3 國家地區代碼範例

代碼	國家	代碼	國家	代碼	國家
TW	中華民國	SG	新加坡	CN	中國
KR	韓國	US	美國	MO	澳門
JP	日本	RU	俄羅斯	HK	香港

- 機場代碼為 IATA 規定，用於代表全世界大多數機場的代碼。它的編號規則為用 3 個字母組成。這些代碼的分配由國際航空運輸協會第 763 號決議決定，並且由在蒙特婁的國際航空運輸協會總部管理。這些代碼每半年一次出版在 IATA 的航空公司編碼目錄裡。因為票務系統以及鐵路航空聯運的需求，國

國際航空運輸協會也為火車站和機場提供代碼，如下表 2-4 所示。

表 2-4 機場 IATA 代碼範例表

代碼	機場	代碼	機場	代碼	車站
ANC	Anchorage	BKK	Bangkok	QQK	Kings Cross Railway Station
CGK	Jakarta	HKG	Hong Kong	XLQ	Guildwood Railway Station
TPE	Taipei	SZX	Shenzhen	QQW	Waterloo Railway Station

## 2.3 關聯式資料庫綱要

關聯式資料庫(Relational Database)是最被廣為使用的資料維護軟體，主要原因是其可以使用 SQL(Structured Query Language)進行各種不同的查詢，且提供快速的查詢處理，以及方便的資料維護介面。因此，在本研究中，我們規劃了一個關聯式資料庫以便處理大量的旅行路徑資料。表格定義如表 2-5 所示，我們設計四個資料庫表格以及其所定義的屬性，其中以底線標示的為主鍵。

首先表格 route(路徑資料) 主要是記錄旅客旅行路徑的資料，有資料年份(year)、啟程機場(orig)、第一轉機機場(stop1)、第二轉機機場(stop2)、第三轉機機場(stop3)、目的機場(dest)以及旅客數(BSPPax)，其中年份、啟程機場與目的機場不可無值，而中轉機場資料若為雙橫槓(--)，則代表無資料，其資料來自於表 2-2 的原始資料。其次，表格 airport(機場資料)主要記錄機場代碼(code)、機場名稱(name)、所在國家(country)、所在的經度(x)、所在的緯度(y) 以及所在的分群(zone)，code 作為 primary key，此表格的目的是在於記錄機場的地理位置。

表 2-5 資料庫綱要

route (路徑資料)		
<u>seq</u>	int	Primary key 記錄第幾年的資料
year	int	記錄啟程機場
orig	varchar(3)	記錄第一轉機機場
stop1	varchar(3)	記錄第二轉機機場
stop2	varchar(3)	記錄第三轉機機場
stop3	varchar(3)	記錄目的機場
dest	varchar(3)	記錄旅客數
BSPPax	int	
airport(機場資料)		
<u>code</u>	varchar(3)	Primary key，記錄機場代碼
name	varchar(100)	記錄機場名稱
country	varchar(2)	記錄機場所在的國家名稱

x	float	記錄機場所在的經度
y	float	記錄機場所在的緯度
zone	varchar(3)	記錄所在地小分群
zone(分群資料)		
<u>code</u>	varchar(3)	Primary key , 紀錄分群名稱
x	float	紀錄分群的平均緯度
y	float	紀錄分群的平均經度
zone_range(分群範圍資料)		
<u>seq</u>	int	Primary key
smallzone	varchar	小分群 , Foreign key (zone.code)
mediumzone	varchar	中分群
largezone	varchar	大分群

注意到 airport 表格中的 zone 欄位，和另兩個表格 zone 與 zone\_range，將機場依照其所在的真實位置進行分群，如此可方便我們取出位於特定地理位置的機場或航線資料進行後續處理。分群方式為先將地圖依七大洲進行『大分群』，再將各大洲所包含的範圍依照聯合國地理區<sup>4</sup>進行『中分群』，如圖 2-2，最後再將各個中分群內的國家稱作『小分群』。不過美國與中國為特例，是因為其國土覆蓋的範圍太大且在旅客旅行路徑中所佔有機場數也很多，所以此兩國家依照一般常見的地理分區法和離本土太遠進行小分群，如下圖 2-3。在表中我們顯示各個分群記錄於資料庫中的代碼，並標示其中之意義。如此分群後，就可以查詢特定範圍的資料，例如大分群可以查詢台灣到美洲，中分群則可以查詢台灣到東歐，小分群可以查詢台灣到中國華東等，各個分群的關係詳見表 2-6。

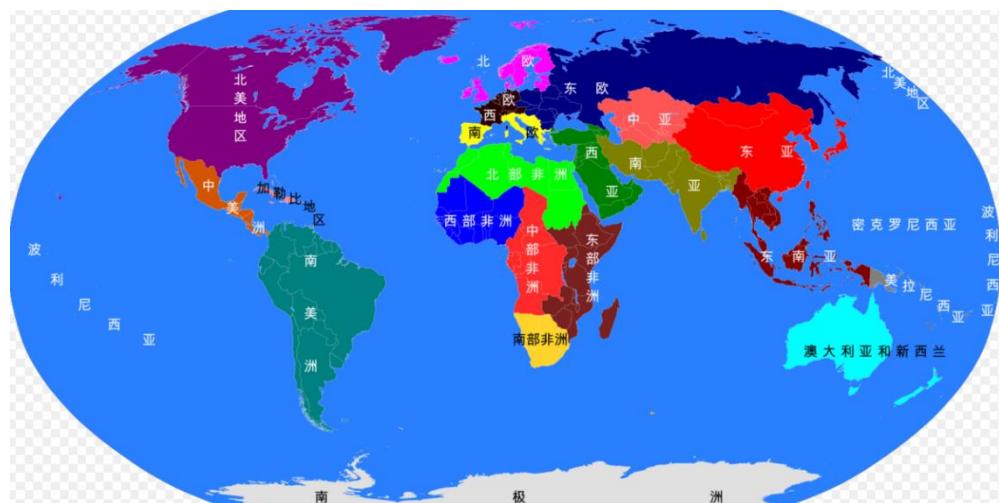
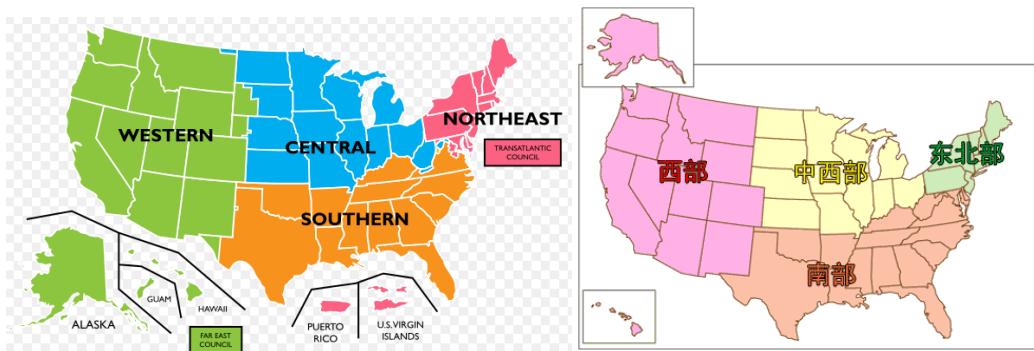


圖 2-2 聯合國地理區

4

<https://zh.wikipedia.org/wiki/%E8%81%94%E5%90%88%E5%9B%BD%E5%9C%B0%E7%90%86%E5%CBA%E5%88%92%E5%88%97%E8%A1%A8>，聯合國地理區劃列表



(a) 美國(本土 4 區加上夏威夷與阿斯維加斯共 6 區)



(b) 中國(本土 7 區加上香港與澳門共 9 區)

圖 2-3 小分群特例

表 2-6 分區對應表(國家為複數集合)

大分群	中分群	小分群
AFC (非洲)	EAF (東非)	國家
	WAF (西非)	國家
	SAF (南非)	國家
	NAF (北非)	國家
	CAF (中非)	國家
EUC (歐洲)	EEU (東歐)	國家
	WEU (西歐)	國家
	SEU (南歐)	國家
	NEU (北歐)	國家
	CEU (中歐)	國家

ASC (亞洲)	EAS (東亞)	中國(9 區)
		國家
	WAS (西亞)	國家
	SAS (南亞)	國家
	NAS (北亞)	國家
	CAS (中亞)	國家
OAC (大洋洲)	AOA (澳大利亞)	國家
	IOA (密克羅尼西亞)	國家
	MOA (美拉尼西亞)	國家
	POA (波里尼西亞)	國家
NAC (北美洲)	NNA (北美大陸)	美國(5 區)
		國家
	MNA (中美地峽)	國家
SAC (南美洲)	CNA (加勒比海群島)	國家
	ESA (東南美)	國家
	WSA (西南美)	國家
	SSA (南南美)	國家
	NSA (北南美)	國家
	CSA (中南美)	國家

## 2.4 資料前置處理

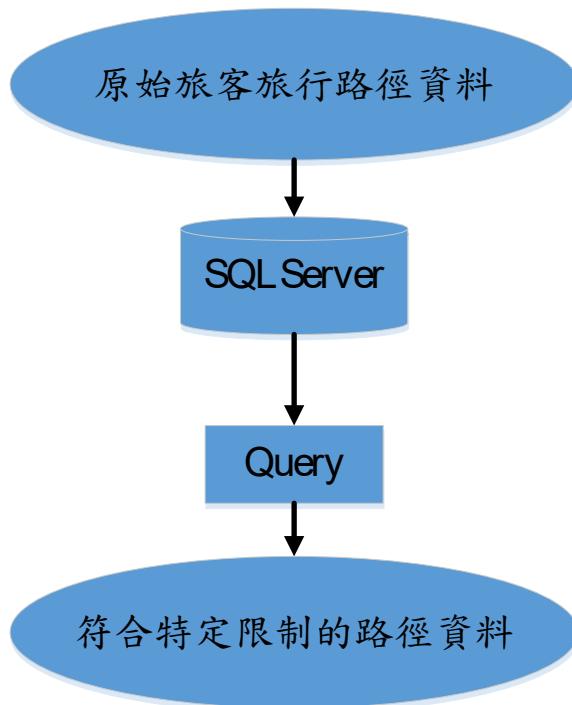


圖 2-4 資料前置處理階段架構圖

本論文提出的前置處理架構如圖 2-4 所示，首先將原始旅客旅行路徑依照 2.3 節的格式，建立於 SQLServer 資料庫中，並利用 Query 得到符合特定限制的旅行路徑資料。例如我們要取出 2013 年臺灣桃園機場至歐洲的旅客旅行路徑資料，Query 範例如下表 2-7 所示。

表 2-7 Query 範例

範例：2013 年臺灣桃園機場至歐洲的旅客旅行路徑並過濾無人使用的路徑	
L01	WITH line AS
L02	(SELECT DISTINCT OrigAirport AS o, Stop1 AS s1, Stop2 AS s2, Stop3 AS s3, DestAirport AS d
L03	FROM route
L04	INNER JOIN airport AS a ON DestAirport = a.code
L05	INNER JOIN zone_range AS zr ON a.zone = zr.smallzone
L06	WHERE (YEAR = 2013) AND (OrigAirport = 'TPE') AND (zr.largezone = 'EUC') AND (BSPPax > 0))
L07	SELECT DISTINCT o.y, o.x, s1.y, s1.x, s2.y, s2.x, s3.y, s3.x, d.y, d.x
L08	FROM line
L09	INNER JOIN airport AS o ON line.o = o.code

L10	INNER JOIN airport AS d ON line.d=d.code
L11	LEFT OUTER JOIN airport AS s1 ON line.s1= s1.code
L12	LEFT OUTER JOIN airport AS s2 ON line.s2= s2.code
L13	LEFT OUTER JOIN airport AS s3 ON line.s3= s3.code;

以下說明此 SQL 查詢句。在表中的 L01 至 L06，首先建立 line 檢視表(View)連結表 2-5 的 3 個資料表，也就是 route、airport 與 zone\_range，以取出符合限定條件的路徑資料。在此範例中，條件為 2013 年的路徑資料中的起程機場為臺灣桃園機場(TPE)且目的機場的所在分群為歐洲(EUC)，並過濾無人使用的路徑，如 L06 Where 子句中的 4 個條件所示。最後於 L02 輸出這些路徑經過的機場代碼並刪除重複路徑，部分範例資料如圖 2-5(a)所示。然後我們在 L07 至 L13 將 line 5 個欄位中的機場代碼依序替換成座標，在 L11 至 L13 使用 LEFT OUTER JOIN 的原因是因為中轉機場欄位值可能為“--”，無法連接 airport 表格，但希望能以 NULL 的方式記錄而不是剔除整筆航線資料，至於起迄機場不會無值，所以使用 INNER JOIN，最後部分的 SQL 結果如下圖 2-5(b)。

	o	s1	s2	s3	d
1	TPE	--	--	--	FRA
2	TPE	--	--	--	CDG
3	TPE	--	--	--	VIE
4	TPE	--	--	--	LHR
5	TPE	HKG	--	--	LHR

(a) line

	oy	ox	sly	s1x	s2y	s2x	s3y	s3x	dy	dx
1	25.0764	121.224	NULL	NULL	NULL	NULL	NULL	NULL	41.8044	12.2508
2	25.0764	121.224	NULL	NULL	NULL	NULL	NULL	NULL	48.0637	16.3411
3	25.0764	121.224	NULL	NULL	NULL	NULL	NULL	NULL	49.0097	2.59777
4	25.0764	121.224	NULL	NULL	NULL	NULL	NULL	NULL	50.0489	8.57056
5	25.0764	121.224	NULL	NULL	NULL	NULL	NULL	NULL	51.2839	-0.2741
6	25.0764	121.224	NULL	NULL	NULL	NULL	NULL	NULL	52.1829	4.4551
7	25.0764	121.224	-6.0725	106.394	24.2559	54.3904	NULL	NULL	37.9364	23.9445

(b) 替換成座標後的 SQL 結果

圖 2-5 資料前處理階段輸出範例

## 第 3 章 系統架構與資料前處理說明

在本章中，我們介紹系統目的與架構，並說明如何讀取路徑資料與切割 cell 範圍。

### 3.1 系統架構

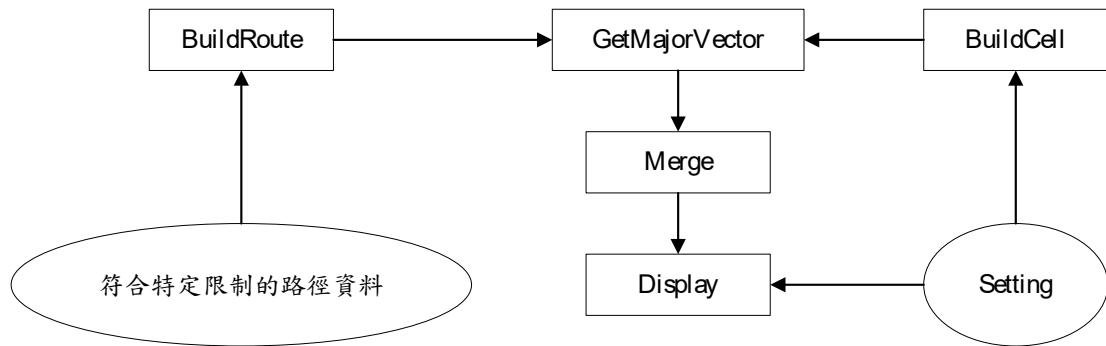


圖 3-1 系統架構圖

本論文的基本想法是將地球切成一塊塊 cell，並將旅客路徑依照 cell 各自求取摘要路徑，最後再加以整合，整個系統架構如圖 3-1 所示。注意到，設定檔『Setting.txt』裡面的設定值決定了『BulidCell 模組』的 cell 範圍大小、路徑資料的系統路徑與『Display 模組』結果顯示的差異門檻。首先，『BulidCell 模組』會依據設定值切割分配 cell 範圍，『BulidRoute 模組』會讀取所有的旅客旅行路徑資料；接著『GetMajorVector 模組』計算 cell 與路徑的交點，並輸出每個 cell 的平均線段、cell 間路徑通過的關係、路徑通過所有 cell 編號與 cell 中不同斜率路徑是否連接，並呼叫『Merge 模組』，依照上述資料連接平均線段；最後，『Display 模組』為了能夠讓使用者便於觀看最後輸出的路徑，會以網頁方式呈現。以下為本研究相關的基本定義：

**[定義 3-1]** 交點：指線與線相交的點，此處的『線』有可能是一個 cell 的邊線。

**[定義 3-2]** 線段：如果航段與 cell 相交，處於 cell 內的航段，我們稱為線段。

**[定義 3-3]** 端點：線段兩端的點。

**[定義 3-4]** 平均線段：將每個 cell 內的線段(端點)依據各個方向平均後的結果，我們稱為平均線段。

### 3.2 BulidCell 模組

本模組的目的是將整個地球(經度範圍-180 至 180、緯度範圍-90 至 90)切成一塊塊相同大小的網格(cell)，並記錄每個網格對應的經緯度範圍。本模組的輸入為經度大小(寬度)和緯度大小(高度)，而輸出為一個 cell 二維矩陣，其維度分別為  $m$  列 (row)、 $n$  行 (column)，每一個元素有  $\text{max}_x$ 、 $\text{min}_x$ 、 $\text{max}_y$ 、 $\text{min}_y$  等 4 個屬性，用來代表 cell 矩形的四個點，也就是：左上( $\text{min}_x, \text{max}_y$ )、左下( $\text{min}_x, \text{min}_y$ )、右上( $\text{max}_x, \text{max}_y$ )以及右下( $\text{max}_x, \text{min}_y$ )，如圖 3-2 範例所示。注意到，在本論文中，經度以變數  $x$  表示，緯度以變數  $y$  表示。

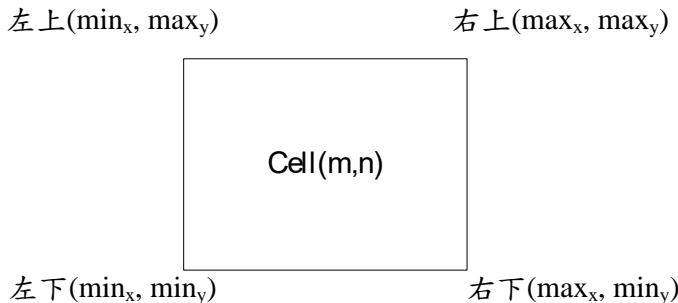


圖 3-2 cell 範例圖

Buildcell 模組程式碼如圖 3-3 所示，根據所指定之經度範圍  $\text{cut}_x$ ，緯度範圍  $\text{cut}_y$ ，在程式 L01 至 L04 執行迴圈，先從緯度開始切，再切經度。在 L04 內層迴圈中從起始值(-180, -90)開始累加  $y$ ，累加的數值為  $\text{cut}_y$ ，從-90 依序累加到 90，然後在 L02 外層迴圈累加  $x$ ，累加的數值為  $\text{cut}_x$ ，從-180 依序累加到 180。在 L05 至 L08 根據所得的  $x$  值和  $y$  值分配 cell 範圍，並在 L09 與 L011 依序變更二維編號  $m, n$ 。

模組名稱：BuildCell

輸入： $\text{cut}_x$ 、 $\text{cut}_y$ ：設定檔中指定之網格經緯度範圍

變數： $m, n$ ：為 cell 矩陣的計數器

輸出：cell 矩陣

L01	$m \leftarrow 0$
L02	<b>For</b> $x \leftarrow -180$ to 180 step $\text{cut}_x$ <b>do</b>
L03	$n \leftarrow 0$
L04	<b>For</b> $y \leftarrow -90$ to 90 step $\text{cut}_y$ <b>do</b>
L05	$\text{cell}[m, n].\text{min}_x \leftarrow x$
L06	$\text{cell}[m, n].\text{max}_x \leftarrow x + \text{cut}_x$
L07	$\text{cell}[m, n].\text{min}_y \leftarrow y$
L08	$\text{cell}[m, n].\text{max}_y \leftarrow y + \text{cut}_y$
L09	$n++$

L10	<b>end for</b>
L11	$m++$
L12	<b>end for</b>

圖 3-3 BuildCell 模組

假設  $\text{cut}_x$  和  $\text{cut}_y$  分別為 10，則此模組輸出的 cell 矩陣如表 3-1 所示，而其對應的切割圖如圖 3-4 所示。我們可看到左下角  $\text{cell}[0, 0]$  對應到南極洲，而台灣大概位於  $\text{cell}[30, 11]$  所代表的經緯度範圍內。

表 3-1 cell 矩陣範例

cell					
註標		元素			
$m$	$n$	$\text{min}_x$	$\text{min}_y$	$\text{max}_x$	$\text{max}_y$
0	0	-180	-90	-170	-80
0	1	-180	-80	-170	-70
.	.	.	.	.	.
1	0	-170	-90	-160	-80
.	.	.	.	.	.
30	11	120	20	130	30
.	.	.	.	.	.

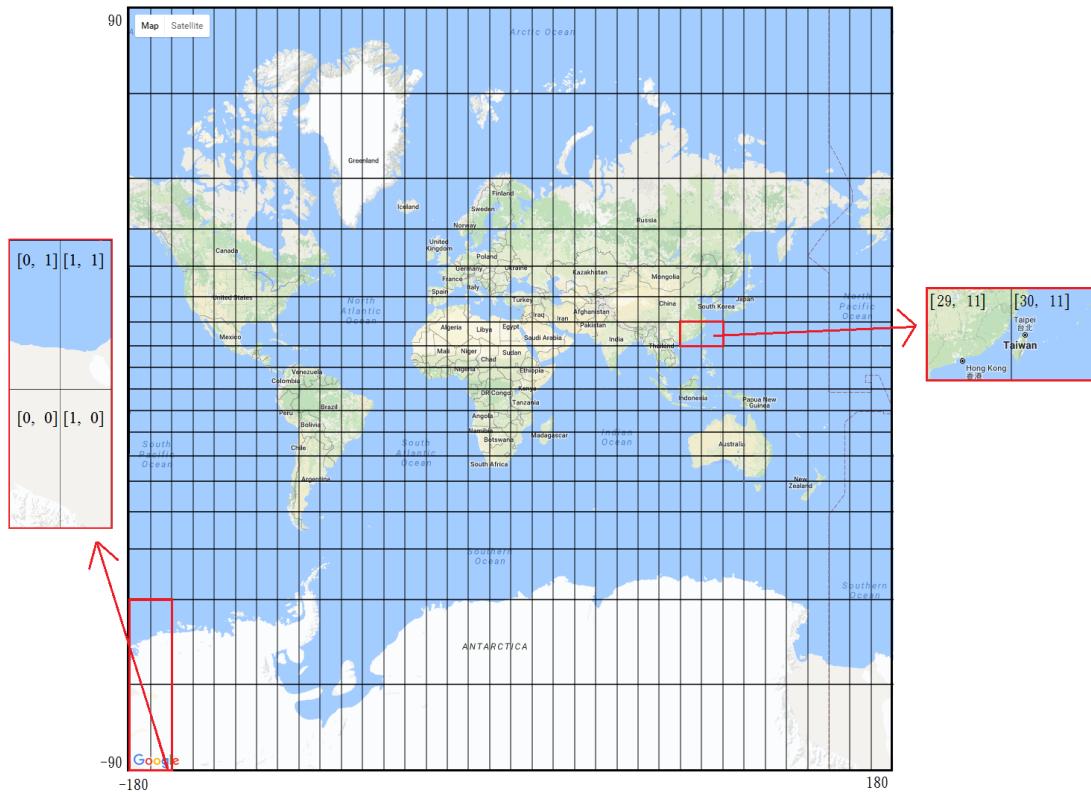


圖 3-4 地球切割圖

注意到，地球是三維的橢球體，而轉換出來的地圖是平面的，所以繪製地圖時會存在將曲面上的形象展現在平面上的問題。本論文使用 Google Map 的 Polyline 物件繪製每個 cell 範圍，而 Google Map 為了方便將世界上位置轉譯成地圖上位置，所以使用麥卡托投影法(Mercator projection)<sup>5</sup>。這是一種等角的圓柱形地圖投影法，但會使投影後的面積產生變形，所以圖 3-4 中的每個方格看起來不一樣大。

### 3.3 BulidRoute 模組

本模組讀取原始的旅客路徑資料，然後建立於內部的 route 二維矩陣，表 3-2 為路徑資料檔範例，一行代表一條旅行路徑所經過的機場，依序為起點、轉機點至迄點，每點記錄該機場之緯度和經度，中間以逗號隔開，若無值則以 NULL 表示。表 3-3 為對應表 3-2 之 route 二維矩陣範例，註標有  $i$  列 (row) 為路徑總數， $j$  行 (column) 對應到構成此路徑的航段，每個元素記錄該航段的起迄點機場座標， $x1$ 、 $x2$  代表經度， $y1$ 、 $y2$  代表緯度。

表 3-2 路徑資料檔(pathfile)範例

行數	旅客旅遊路徑資料
1	25.0764,121.224,NULL,NULL,NULL,NULL,NULL,NULL,49.205,119.825
2	25.0764,121.224,22.3089,113.914,-37.0029,174.473,-43.4894,172.532,-45.9281,170.198

表 3-3 route 矩陣範例

route					
註標		元素			
$i$	$j$	$x1$	$y1$	$x2$	$y2$
1	1	121.224	25.0764	119.825	49.205
2	1	121.224	25.0764	113.914	22.3089
	2	113.914	22.3089	174.473	-37.0029
	3	174.473	-37.0029	172.532	-43.4894
	4	172.532	-43.4894	170.198	-45.9281

BulidRoute 模組如圖 3-5 所示，在程式 L01 讀取每筆旅客旅行路徑資料，在 L02 與 L03 將路徑中之航段依序儲存至矩陣 route 中。注意到於 L04 行我們傳入每筆航段的 route 矩陣至 adjust 演算法中，判斷是否需要調整變動該筆 route 矩陣，這是因為我們希望連接該航段起點( $x_1, y_1$ )和迄點( $x_2, y_2$ )的直線，代表到( $x_1, y_1$ )和

<sup>5</sup>

<https://zh.wikipedia.org/wiki/%E9%BA%A5%E5%8D%A1%E6%89%98%E6%8A%95%E5%BD%B1%E6%B3%95>，麥卡托投影法

$(x_2, y_2)$ 的最短距離。由於地球是球形，所以上述現象不一定為真。舉例來說，如圖 3-6(a)所示，如果有一航段從點 1(120, 25)移動至點 2(-160, 25)，以二維平面來說直接連接該兩點的航段如圖 3-6(a)橫跨 280 經度，但實際的最短路徑如圖 3-6(b)，這也就是要繞行地球的另一邊，橫跨 80 經度就好。所以為了能夠將連接點 1 和點 2 的最短路徑記錄於 route 中，以作為後續的計算，就需要將該航段切割成兩小段，如圖 3-6(b)所示。

模組名稱：BulidRoute

輸入：pathfile 為旅遊路徑資料檔

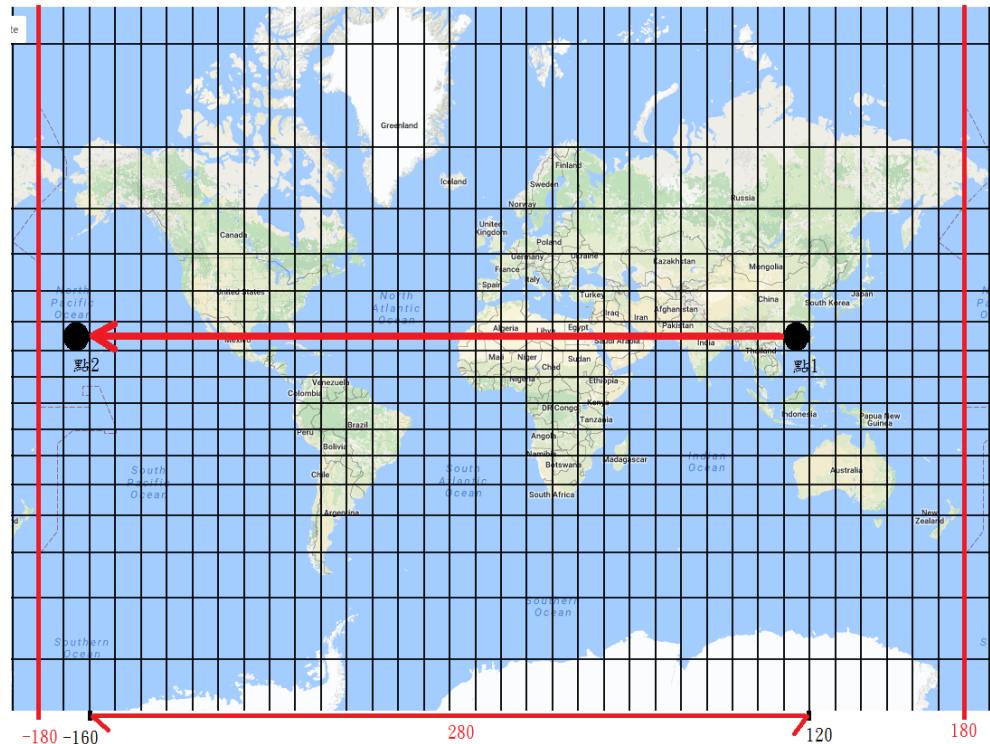
變數： $i$ ：路徑的計數器

$j$ ：航段的計數器輸出

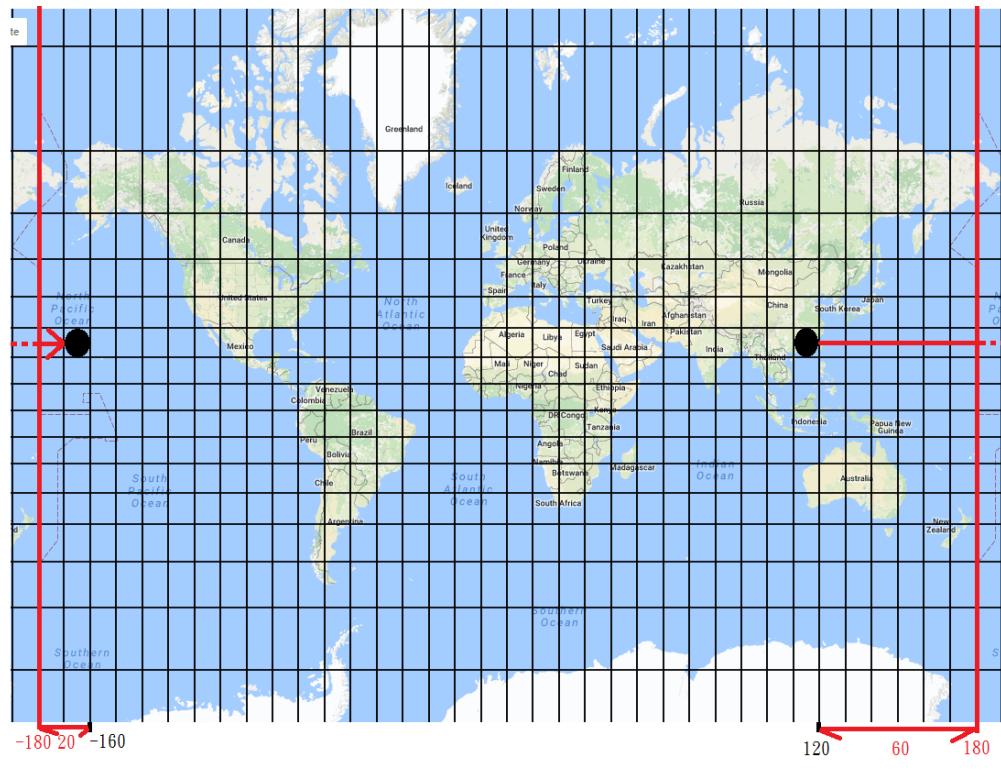
輸出：route 矩陣

L01	<b>for each</b> line[ $i$ ] in pathfile
L02	<b>for each</b> (segment[ $j$ ] in line[ $i$ ])
L03	route[ $i, j$ ].x1、route[ $i, j$ ].y1、route[ $i, j$ ].x2、route[ $i, j$ ].y2 $\leftarrow$ segment[ $j$ ]
L04	route = adjust(route[ $i, j$ ])
L05	<b>end for</b>
L06	<b>end for</b>

圖 3-5 BulidRoute 模組



(a)



(b)

圖 3-6 調整航段的範例圖

adjust 演算法如圖 3-7 所示，在 L01 中我們先計算兩點間經度的差距，如果當其超過了二維平面的一半，如圖 3-6(a)，走另一邊會比較短，如圖 3-6(b)，所以我們必須對該航段進行變換與切割。切割前必須對端點座標進行變換，是因為未變換前的航段，如圖 3-6(a)並不會與地球邊界相交，所以必須將點 2 的座標變換成相應的二維座標，在這裡地圖的邊界(border)有一樣的情況。在 L03 中我們呼叫 extend 函數，依照以下規則得到變換後的點 2 座標(exd2, exdy2)與地圖的邊界(border)：我們依照點 1 所在的經度正負值，決定對應的二維平面座標往左(往經度負值移動)還是往右延長(往經度正值移動)，如果往右走，那就將點 2 座標經度加 360 度(移到點 1 的右邊)，且將 border 設定為正的(180, 90, 180, -90)；如果往左走，那就將點 2 座標經度減 360 度(移到點 1 的左邊)，且將 border 設定為負的(-180, 90, -180, -90)。

接著，在 L04 中使用 intersect 演算法計算航段與地圖邊界的交點，並在 L06 到 L09 將點 1 到交點視為第一段分割航段，交點到點 2 視為第二段分割航段。請注意這裡一開始所計算出的交點(node1)，彼此關係為點 1(x1, y1)→交點(node1)→變換後的點 2(x2±360, y2)，但我們儲存至矩陣 route 的點 2 必須為原始座標，所以我們需要再次轉換交點(node1)的經度值以對應點 2，並在 L05 使用 checknode 函數將其轉換為 node2。最後儲存至 route 矩陣結果為兩個航段，分別為點 1 → node1, node2 → 點 2，在此 checknode 函數的規則為輸入 node1(nx, ny) 與點 2(x2, y2)，此處的 nx 可能為 180 或是-180(交點在地球的邊界上)，然後將

node1 的經度值變換與點 2 一致後回傳為 node2(如果點 2 經度為正，nx 為 180，反之亦然)。至於 L11 則表示航段不通過邊界，不變動矩陣 route 直接回傳。

演算法名稱：adjust 輸入：route[i, j]，由(x1, y1)和(x2, y2)兩點組成 變數：diff：計算兩點之間經度的差異 border：地球的邊界(bx1, by1, bx2, by2) node1, node2：航段與 border 交點座標 extendx2, extendy2：變換後的座標 輸出：route	
L01	diff =   x2 - x1
L02	<b>if</b> (diff > 180) <b>then</b>
L03	extendx2, extendy2, border ← extend(x1, x2, y2)
L04	node1 ← intersect(x1,y1, extendx2, extendy2, border)
L05	node2 ← checknode(x2,y2, node1)
L06	route[i, j].x1, route[i, j].y1 ← (x1,y1)
L07	route[i, j].x2, route[i, j].y2 ← node1
L08	route[i, j + 1].x1, route[i, j + 1].y1 ← node2
L09	route[i, j + 1].x2, route[i, j + 1].y2 ← (x2,y2)
L10	<b>return</b> route
L11	<b>else</b>
L12	<b>return</b> route
L13	<b>end if</b>

圖 3-7 adjust 演算法

我們以圖 3-6 作為 adjust 演算法範例，範例航段為點 1 (120, 25)→點 2(-160, 25)，由於點 1 經度為正值，所以航段的最短路徑為往右移動通過經度為正的 border (180, 90, 180, -90)，且點 2 轉換後的座標為(200, 65)。呼叫 intersect 演算法找出航段與 border 的交點為(180, 25)，然後使用 checknode 函數將交點(180, 25)轉換為(-180, 25)以便對應點 2，最後回傳的矩陣結果為點 1(120,25)→交點 node1(180,25)， node2 (-180,25)→點 2(-160,25)，實際內容如表 3-4。

表 3-4 adjust 範例

route					
註標		元素			
i	j	x1	y1	x2	y2
1	1	120	25	180	25
	2	-180	25	-160	25

### 3.4 計算交點演算法

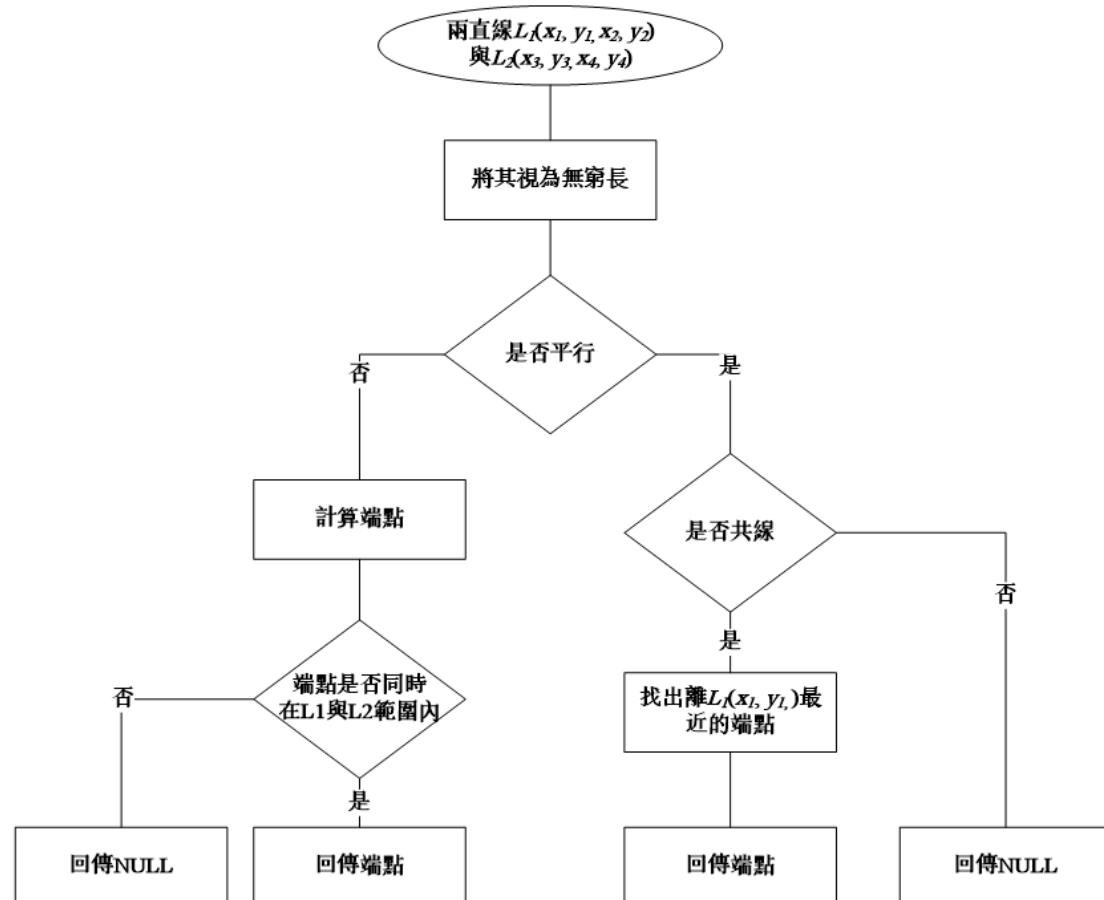


圖 3-8 求取兩直線交點的流程圖

本節說明 `adjust` 演算法(圖 3-7)所呼叫的 `intersect` 演算法，其流程圖如圖 3-8 所示。計算線段  $L_1$  與線段  $L_2$  相交的交點  $P(P_x, P_y)$ 方法，首先將兩線段視為無窮長，使用外積<sup>6</sup>(也稱叉積)公式(1)，判斷兩線段彼此是平行還是相交。如果  $\sin\theta$  值不為零，則兩線相交再使用直線方程式<sup>7</sup>(2)計算交點  $P$ ，也就是使用兩點式求  $L_1$  與  $L_2$  的聯立解。然後，使用內積<sup>8</sup>(也稱點積)公式(3)檢查交點  $P$  是否在線段  $L_1$  上，如為是則回傳交點  $P$ ，如為否則回傳不相交。如為平行，則使用法向量<sup>9</sup>(垂直該直線(或平面)的向量)的平行線距離公式(4)計算兩線段之間的距離，判斷是否為共線，如為是則表示  $L_1$  與  $L_2$  重疊，將  $L_2$  兩端點中距離  $L_1$  端點  $(x_1, y_1)$  最近的點作為交點並回傳，如為否則回傳不相交。請注意四個公式中的變數與數學式都是共用的。

<sup>6</sup> <https://zh.wikipedia.org/wiki/%E5%A4%96%E7%A7%AF>，外積

<sup>7</sup> <https://zh.wikipedia.org/wiki/%E7%9B%B4%E7%BA%BF>，直線方程式

<sup>8</sup> <https://zh.wikipedia.org/wiki/%E6%95%B0%E9%87%8F%E7%A7%AF>，內積

<sup>9</sup> <https://zh.wikipedia.org/wiki/%E6%B3%95%E7%BA%BF>，法向量

$$L1 = (x_1, y_1), (x_2, y_2) , L2 = (x_3, y_3), (x_4, y_4)$$

$$A1 = y_2 - y_1 , A2 = x_2 - x_1$$

$$B1 = y_4 - y_3 , B2 = x_4 - x_3$$

$$C1 = (x_2 \times y_1) - (x_1 \times y_2) , C2 = (x_4 \times y_3) - (x_3 \times y_4)$$

$$Cross(L1 \times L2) = |L1||L2|\sin\theta = ((A1 \times B2) - (A2 \times B1)) \quad (1)$$

$$\text{Pythagorean(兩點式)} \begin{cases} L1 : Py = \frac{(Px - x_1) \times A1}{A2} + y_1 \\ L2 : Py = \frac{(Px - x_3) \times B1}{B2} + y_3 \end{cases}$$

$$Px = \frac{(B2 \times C1) - (B1 \times C2)}{(A2 \times B1) - (A1 \times B2)}, Py = \frac{(A1 \times C2) - (A2 \times C1)}{(A2 \times B1) - (A1 \times B2)}$$

(2)

$$Dot(P, L1) = (x_1 - Px) \times (x_2 - Px) + (y_1 - Py) \times (y_2 - Py) \quad (3)$$

$$Distance(L1, L2) = \frac{|C1 - C2|}{\sqrt{(A2 + B2)}} \quad (4)$$

完整的 intersect 演算法如圖 3-9 所示，其中分別在 L01 呼叫叉積公式(1)，先判斷線段彼此是平行還是相交，如為平行不相交則在 L02 呼叫平行線距離公式(4)，判斷是否為共線，如為是則在 L03 的 nearby 函數中回傳 L2 的端點中距離 L1 最近的點為交點 P 並回傳，如為否則回傳不相交。如為相交則在 L09 的 Pythagorean 函數使用公式(2)計算交點 P，然後在 L10 至 L12 使用點積公式(3)，如回傳的數值小於或等於 0，則表示交點 P 在線上並回傳，如為否則回傳不相交。用圖 3-6 作範例說明，首先航段為(120,25,200,25)，地球邊界為(180,90,180,-90)，在 L01 中點積值為 12 不為 0，所以在 L09 中計算出交點 P(180,25)，然後在 L10 中回傳值為-120 與-7475 都小於 0，所以最後在 L11 回傳交點 P。

演算法名稱：intersect

輸入：L1(x<sub>1</sub>, y<sub>1</sub>, x<sub>2</sub>, y<sub>2</sub>), L2(x<sub>3</sub>, y<sub>3</sub>, x<sub>4</sub>, y<sub>4</sub>)

輸出：P：L1 與 L2 的交點

L01	<b>if</b> Cross ( L1, L2 ) = 0 <b>then</b> //平行
L02	<b>if</b> Distance ( L1, L2 ) = 0 <b>then</b> //共線
L03	P = nearby ( x <sub>1</sub> , y <sub>2</sub> , L2 )
L04	<b>return</b> P
L05	<b>else</b> //不共線，永不相交

L06	<b>return</b> null
L07	<b>end if</b>
L08	<b>else</b> //相交
L09	$P = \text{Pythagorean}(L1, L2)$
L10	<b>if</b> Dot( $P, L1$ ) $\leq 0$ & Dot( $P, L2$ ) $\leq 0$ <b>then</b> //交點在 $L1$ 與 $L2$ 上
L11	<b>return</b> $P$
L12	<b>else</b> //交點不在 $L1, L2$ 上
L13	<b>return</b> null
L14	<b>end if</b>
L15	<b>end if</b>

圖 3-9 intersect 演算法

## 第 4 章 航線整合與顯示

在本章中，我們說明如何將 BuildRoute 模組所回傳的所有航段依照 BuildCell 模組所計算的 cell 範圍分段並摘要化。我們會介紹所用的矩陣結構，以及遇到的問題和解決方法。

### 4.1 範例說明

在接下來的章節，我們使用表 4-1 的完整範例資料進行說明，此範例包含 4 條航線，涵蓋 3 個 cell 大小，表 4-1(a)為原始航線資料，表 4-1(b)為 route 矩陣內容，表 4-1(c)為 cell 矩陣中，路徑經過的 3 個 cell 範圍值。圖 4-1 為未處理前的範例圖，可看到這 4 條航線為從台灣出發，分別在香港與澳門(各 2 條)中轉至不同地方，其中編號 1 和 2 的航線在澳門轉機之後分別抵達長沙市與重慶市，編號 3 和 4 的航線在香港轉機之後分別抵達北海市與南寧市。

表 4-1 完整範例資料

(a) 原始航線資料範例

行數	旅客旅遊路徑資料
1	25.0764,121.224,22.0858,113.353,NULL,NULL,NULL,NULL,28.1124,113.131
2	25.0764,121.224,22.0858,113.353,NULL,NULL,NULL,NULL,29.4312,106.382
3	25.0764,121.224,22.3089,113.914,NULL,NULL,NULL,NULL,21.5394,109.294
4	25.0764,121.224,22.3089,113.914,NULL,NULL,NULL,NULL,22.3636,108.102

(b) route 矩陣範例

route					
註標		元素			
$i$	$j$	x1	y1	x2	y2
1	1	121.224	25.0764	113.353	22.0858
	2	113.353	22.0858	113.131	28.1124
2	1	121.224	25.0764	113.353	22.0858
	2	113.353	22.0858	106.382	29.4312
3	1	121.224	25.0764	113.914	22.3089
	2	113.914	22.3089	109.294	21.5394
4	1	121.224	25.0764	113.914	22.3089
	2	113.914	22.3089	108.102	22.3636

(c) 航線經過的 cell 範圍

cell					
註標		註標			
$m$	$n$	$\text{max}_x$	$\text{min}_x$	$\text{max}_y$	$\text{min}_y$
28	11	110	100	30	20
29	11	120	110	30	20
30	11	130	120	30	20



圖 4-1 未處理前的範例示意圖

## 4.2 GetMajorVector 模組介紹與 avgcell 說明

『GetMajorVector 模組』的目的為將 route 矩陣與 cell 矩陣相交，以取得所有 cell 中的平均線段。接下來我們先介紹儲存平均線段的 avgcell 矩陣結構。avgcell 矩陣的功能為加總線段端點，下表 4-2 為 avgcell 的矩陣結構，註標中的  $m$ 、 $n$ ，對應 cell 矩陣大小，編號  $d(\text{direction})$  則表示航段方向，共 8 種，od(起迄)記錄平均線段的起點座標( $\text{avgcell}[m, n, d, 0]$ )與迄點座標  $\text{avgcell}[m, n, d, 1]$ )。而元素內記錄的  $\text{total}_x$ 、 $\text{total}_y$  為加總後的線段端點座標，count 則記錄矩陣中存入了幾個端點，以便最後計算平均值。

表 4-2 avgcell 矩陣結構

avgcell							
註標				元素			
欄位名	$m$	$n$	$d$	od	$\text{total}_x$	$\text{total}_y$	count
說明	對應 cell 座標 m	對應 cell 座標 n	航段 方向	起迄 點	加總端點 座標經度	加總端點 座標緯度	端點總數

註標  $d(\text{direction})$  表示航段方向，我們將航段  $(x_1, y_1, x_2, y_2)$  以 8 種方向表示，如圖 4-2，計算方法為將航段的兩點座標，帶入直線方程式中求斜率，當斜率大於 0 時，航段為左下往右上傾斜，斜率小於 0 時，航段為左上往右下傾斜，斜率

等於 0，航段平行 x 軸，斜率不存在時，航段垂直 x 軸，最後依照上下方向給予其編號，如表 4-3。

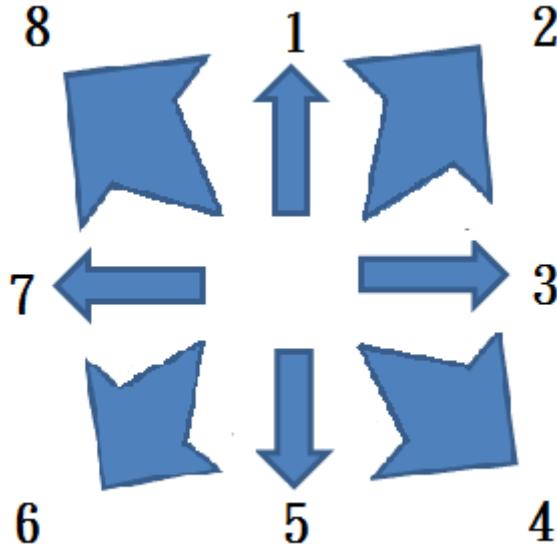


圖 4-2 航段方向種類範例

表 4-3  $d(\text{direction})$  表示航段方向

斜率	direction	方向性	判斷
不存在	1	上	$y_1 < y_2$
	5	下	$y_1 > y_2$
大於 0	2	右上	$x_1 < x_2 \& y_1 < y_2$
	6	左下	$x_1 > x_2 \& y_1 > y_2$
等於 0	3	右	$x_1 < x_2$
	7	左	$x_1 > x_2$
小於 0	4	右下	$x_1 < x_2 \& y_1 > y_2$
	8	左上	$x_1 > x_2 \& y_1 < y_2$

GetMajorVector 模組如圖 4-3 所示，首先，L01 到 L05 表示依序讀取 route 矩陣資料，L07 取得包含該航段的最小包圍矩形(minimum bounding rectangle)，我們稱為 MBR，然後依據表 4-3，在 L08 判斷航段的方向取得  $d$ 。而在目前的做法之中我們發現了一個問題，也就是我們將航段與 cell 邊界相交的交點儲存至 avgcell 矩陣中，但當起迄點不在 cell 邊界上的情況下，也就會因為彼此不相交就不會儲存，可是這就會導致航段變短了，所以我們在 L11 到 L22 利用 onedge 函數分別判斷起迄點是否在 cell 邊界上，如果為否則將其儲存至 avgcell 矩陣中，如為是就不儲存。然後我們介紹 madeline 演算法中所使用的變數 front、checkm、checkn 和 Set 集合，其中 front 用來記錄前一段航段的方向  $d$ ，而 checkm 和 checkn 變數用來記錄 cell 編號，Set 用來儲存與航段相交的 cell 集合。在 L03 與 L04 中，

在每條航線判斷前先設定初始值，front 初始值為 0，checkm 和 checkn 的初始值為  $[-1, -1]$ ，這是由於 cell 編號可以為 0 的關係，並且將 Set 集合每排序完一條航段後在 L06 清空。

其次在 L23 判斷 MBR 是否與 cell 相交，如果相交則在 L24 的 intersect 演算法中，依序計算航段與 cell 的哪一個邊相交並計算交點座標  $P$ ，由於交點數最多為 2，所以分別稱為  $P_1$  與  $P_2$ 。在 L25 至 L33 中，先檢查  $P_i$  是否都為 NULL，如為是則表示航段與該 cell 不相交，如為否則在 L28 使用 checkod 函數判斷  $P_i$  為線段中的起點還是迄點。該函數判斷交點  $P_i$  是在 cell 哪個邊界上(航段進入或離開)，其參數包含航段方向  $d$ 、交點座標與 cell 編號，也就是藉由航段方向  $d$  去檢查座標  $P_i$  在 cell 的哪個邊界上，例如  $d$  為 4(往右下移動)，起點座標會位於 cell 的上與左邊界，迄點座標則位於 cell 的下與右邊界。然後在 L29 和 L30 將  $P_i$  儲存至 avgcell 矩陣中。

在 L35 到 L37 中將與該航段相交的 cell 座標記錄至 Set 中。每當一條航段判斷完後(L39)，就執行 madeline 演算法依照不同航段方向排序 Set，然後回傳 link 矩陣、turn 矩陣與所需的相關變數。我們會在下一節介紹 link 矩陣、turn 矩陣與相關的演算法。

模組名稱：GetMajorVector

輸入：cell 矩陣：儲存切割出來的每個 cell 範圍資料

route 矩陣：儲存旅客旅行路徑之航段資料， $(x_1, y_1)$ 為起點， $(x_2, y_2)$ 為迄點

變數：MBR：代表包含該航段的最小矩形範圍

touch：檢查該 cell 是否碰到航段，以 False 與 True 表示

$d$ ：航段方向

Set：儲存與航段相交的 cell 集合

$P_i$ ：航段與 cell 的交點，最多為 2 個，若回傳 NULL 代表無交點

$P_{i0d}$ ：交點在線段中為起點還迄點，其值為 0 或 1

front：記錄前一個航段方向  $d$ ，初始為 0，為 madeline 演算法所需

checkm, checkn：記錄前一個航段最後移動到的 cell 編號

輸出：avgcell 矩陣：記錄平均線段

link 矩陣：每條航段走的 cell 編號集合

turn 矩陣：記錄每個 cell 中不同方向的相連航線

L01	<b>for each</b> $i$ <b>in</b> route[ $i, j$ ]
L02	$j \leftarrow 0$
L03	$front \leftarrow 0$
L04	$[checkm, checkn] \leftarrow [-1, -1]$
L05	<b>while</b> ( route[ $i, j$ ] != NULL)
L06	$Set \leftarrow \text{NULL}$
L07	$MBR \leftarrow \text{getmbr}(\text{route}[i, j])$

L08	<code><math>d \leftarrow \text{getdir}(\text{route}[i, j])</math></code>
L09	<code><b>for each</b> <math>\text{cell}[m, n]</math> <b>do</b></code>
L10	<code>    touch <math>\leftarrow \text{False}</math></code>
L11	<code>    <b>if</b> (<math>\text{onedge}(\text{cell}[m, n], \text{route}[i, j].x1, \text{route}[i, j].y1) = \text{True}</math>)</code>
L12	<code>        touch <math>\leftarrow \text{True}</math></code>
L13	<code>        <math>\text{avgcell}[m, n, d, 0].\text{total}_x += \text{route}[i, j].x1</math></code>
L14	<code>        <math>\text{avgcell}[m, n, d, 0].\text{total}_y += \text{route}[i, j].y1</math></code>
L15	<code>        <math>\text{avgcell}[m, n, d, 0].\text{count} ++</math></code>
L16	<code>    <b>end if</b></code>
L17	<code>    <b>if</b> (<math>\text{onedge}(\text{cell}[m, n], \text{route}[i, j].x2, \text{route}[i, j].y2) = \text{True}</math>)</code>
L18	<code>        touch <math>\leftarrow \text{True}</math></code>
L19	<code>        <math>\text{avgcell}[m, n, d, 1].\text{total}_x += \text{route}[i, j].x2</math></code>
L20	<code>        <math>\text{avgcell}[m, n, d, 1].\text{total}_y += \text{route}[i, j].y2</math></code>
L21	<code>        <math>\text{avgcell}[m, n, d, 1].\text{count} ++</math></code>
L22	<code>    <b>end if</b></code>
L23	<code>    <b>if</b> (<math>\text{cell}[m, n]</math> intersects MBR)</code>
L24	<code>        <math>P(P_1, P_2) \leftarrow \text{intersect}(\text{cell}[m, n], \text{route}[i, j])</math></code>
L25	<code>        <b>for each</b> <math>P_i</math> in <math>P</math></code>
L26	<code>            <b>if</b> (<math>P_i \neq \text{NULL}</math>)</code>
L27	<code>                touch <math>\leftarrow \text{True}</math></code>
L28	<code>                <math>P_{i\text{od}} \leftarrow \text{checkod}(d, P_i, \text{cell}[m, n])</math></code>
L29	<code>                <math>\text{avgcell}[m, n, d, P_{i\text{od}}].\text{total}_x + \text{total}_y += P_i</math></code>
L30	<code>                <math>\text{avgcell}[m, n, d, P_{i\text{od}}].\text{count} ++</math></code>
L31	<code>            <b>end if</b> //L26</code>
L32	<code>        <b>end for</b> //L25</code>
L33	<code>    <b>end if</b> //L23</code>
L34	<code>    <b>if</b> ( touch = True )</code>
L35	<code>        Set.add(<math>m, n</math>)</code>
L36	<code>    <b>end if</b> //L35</code>
L37	<code>    <b>end for</b> //L09</code>
L38	<code>link, turn, front, checkm, checkn <math>\leftarrow \text{madeline}(\text{front}, \text{checkm}, \text{checkn}, d, \text{Set})</math></code>
L39	<code>    j ++</code>
L40	<code>    <b>end while</b> //L06</code>
L41	<code>    <b>end for</b> //L01</code>

圖 4-3 GetMajorVector 模組

表 4-4 為表 4-1 完整範例資料經過『GetMajorVector 模組』後的結果，此範例中總共 4 條航線，我們以第 1 條航線作為模組範例。第 1 條航段為台灣到澳門，

先在 L07 計算航段 MBR(詳細作法與計算會在下段說明)，然後在 L08 計算出航段方向  $d$  為 6(往左下方移動)，在 L09 中依序讀取 cell 範圍，並在 L11 和 L17 中判斷航段起迄點是否在 cell 的邊界上，首先我們會在 cell[29, 11] 中，檢查到迄點在 cell 內，所以在 L17 到 L22 中將迄點(113.353, 22.0858)存入 avgcell[29, 11, 1, 6] 中，接著在 L23 中得知航段與 cell[29, 11] 相交，所以計算其交點並得到  $P_1$ (120, 24.611339092872)，然後在 L28 使用 checkod 演算法，發現航段方向  $d$  為 6(往左下方移動)，此航段只可能從 cell 的右或上邊界進入，所以當交點  $P_1$  位於 cell[29, 11] 的右邊界上時， $P_{1od}$  為 0(進入點)，然後在 L29 和 L30 存入至 avgcell[29, 11, 0, 6]。我們繼續檢查其他 cell，發現起點在 cell[30, 11] 中，所以在 L11 到 L16 中將起點(121.224, 25.0764)存入 avgcell[30, 11, 0, 6] 中，接著也在 L23 中得知航段與 cell[30, 11] 相交，所以計算其交點並得到  $P_1$ (120, 24.611339092872)，然後在 L28 使用 checkod 演算法，發現交點  $P_1$  位於 cell[30, 11] 的左邊界上， $P_{1od}$  為 1(離開點)，最後將交點  $P_1$  存入 avgcell[30, 11, 1, 6] 中。

接著檢查第 2 條航段為澳門到長沙市，作法與上述相同，但由於此航段與所有的 cell 邊界不相交，所以我們將起迄點處理完就結束了。先計算出航段方向  $d$  為 8，接著在 L12 到 L22 中將起點(113.353, 22.0858)存入 avgcell[29, 11, 0, 8]，迄點(113.131, 28.1124)存入 avgcell[29, 11, 1, 8]，其他航線以此類推。我們於圖 4-4 飄是所有交點。如圖 4-4(a)第 1 條航線共 2 條航段，與 cell 相交後有 6 個交點與 3 條線段，詳細交點座標如表 4-4(a)的編號 1 至編號 6；如圖 4-4(b)第 2 條航線共 2 條航段，與 cell 相交後有 8 個交點與 4 條線段，詳細交點座標如表 4-4(a)的編號 7 至編號 14；如圖 4-4(c)第 3 條航線共 2 條航段，與 cell 相交後有 6 個交點與 3 條線段，詳細交點座標如表 4-4(a)的編號 15 至編號 20；如圖 4-4(d)第 4 條航線共 2 條航段，與 cell 相交後有 8 個交點與 4 條線段，詳細交點座標如表 4-4(a)的編號 21 至編號 28，總共 28 個交點。我們將 avgcell 矩陣中的 total<sub>x</sub> 與 total<sub>y</sub> 除以 count，如表 4-4(b)的 x 和 y 所示，而對應的圖則如圖 4-5 所示。

表 4-4 avgcell 範例

(a)

編號	$m$	$n$	od	$d$	交點經度	交點緯度
1	30	11	0	6	121.224	25.0764
2	30	11	1	6	120	24.6113390928726
3	29	11	0	6	120	24.6113390928726
4	29	11	1	6	113.353	22.0858
5	29	11	0	8	113.353	22.0858
6	29	11	1	8	113.131	28.1124
7	30	11	0	6	121.224	25.0764
8	30	11	1	6	120	24.6113390928726
9	29	11	0	6	120	24.6113390928726

10	29	11	1	6	113.353	22.0858
11	29	11	0	8	113.353	22.0858
12	29	11	1	8	110	25.6188836608808
13	28	11	0	8	110	25.6188836608808
14	28	11	1	8	106.38	29.4312
15	30	11	0	6	121.224	25.0764
16	30	11	1	6	120	24.6130046511628
17	29	11	0	6	120	24.6130046511628
18	29	11	1	6	110	21.6569902597403
19	28	11	0	6	110	21.6569902597403
20	28	11	1	6	109.294	21.5394
21	30	11	0	6	121.224	25.0764
22	30	11	1	6	120	24.6130046511628
23	29	11	0	6	120	24.6130046511628
24	29	11	1	6	113.914	22.3089
25	29	11	0	8	113.914	22.3089
26	29	11	1	8	110	22.3457368547832
27	28	11	0	8	110	22.3457368547832
28	28	11	1	8	108.102	22.3636

(b)

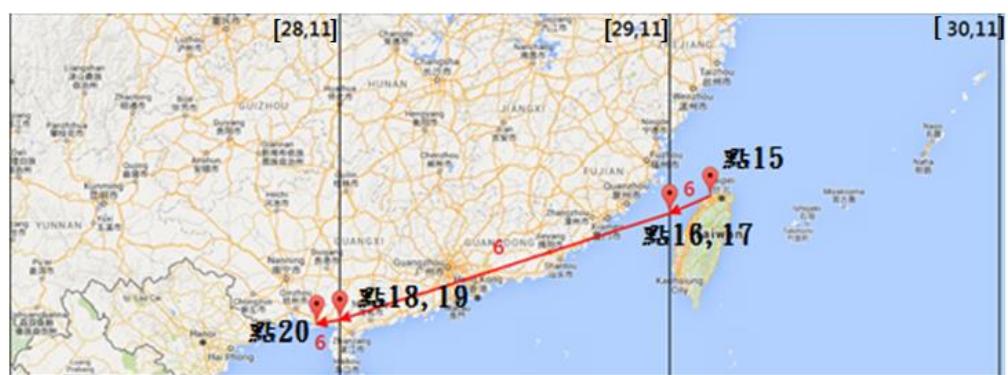
avgcell						
註標				元素(平均後的值)		編號
$m$	$n$	$d$	od	$x$	$y$	
28	11	6	0	110	21.6569902597403	1
			1	109.294	21.5394	
		8	0	110	23.982310257832	2
			1	107.242	25.8974	
29	11	6	0	160	32.81622916269027	3
			1	150.20667	29.37916341991343	
		8	0	113.54	22.160167	4
			1	111.043667	25.35900683855467	
30	11	6	0	121.224	25.0764	5
			1	120	24.6121718720177	



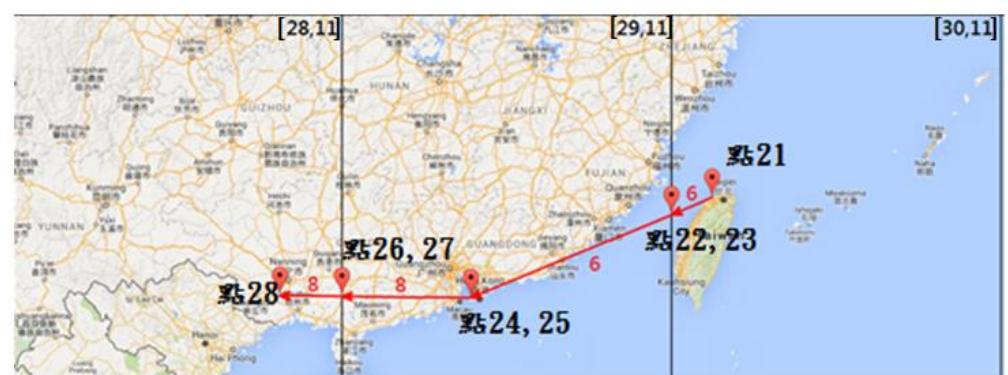
(a)



(b)



(c)



(d)

圖 4-4 完整交點座標範例



圖 4-5 GetMajorVector 模組結果圖

以下解釋為何程式在 L07 先求取 MBR 並於 L23 判斷 cell 是否與該 MBR 相交，而非如 L24 直接求取航段是否與 cell 相交。首先，判斷 MBR 是否與 cell 相交的方法，如下圖 4-6 假設 2 個矩形  $a, b$  的中心分別為  $o_a$  與  $o_b$ ，如果 2 個矩形要相交必須滿足以下 2 個條件： $o_a$  與  $o_b$  橫座標距離小於兩矩形橫邊和的一半，且  $o_a$  與  $o_b$  縱座標距離小於兩矩形縱邊和的一半。我們稱此運算為『MBR 相交檢查』。

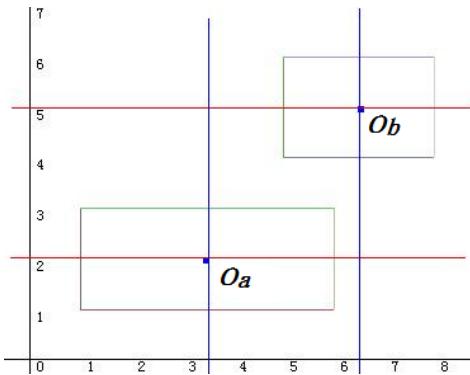


圖 4-6 MBR 相交檢查法

注意到，直接檢查航段與 cell 的 4 個邊界是否相交，所需的公式(2)計算量為 4，以下我們比較是否先用 MBR 的計算量差別。用直接檢查的方式，需要的計算量為 cell 數  $\times$  4 次公式(2)。如果使用 MBR 檢查法，計算量為找出 cell 是否與 MBR 相交(cell 數  $\times$  1 次 MBR 相交檢查)，再加上如果 cell 與 MBR 有相交，再使用公式(2)計算出交點。如表 4-5 我們假設一條航段橫跨 3 個 cell，可以發現 MBR 檢查法只使用公式(2)12 次，整體計算量比直接檢查法少很多。以本論文分析的航線資料而言，大部分的航段所橫跨的 cell 數都很少，所以整體而言可以減少大量的計算次數。就算我們以 worst case 來討論，例如在 648 個 cell 中，航段需要橫跨 486 個 cell 以上，MBR 檢查法的計算量才會超過直接檢查法，但在真實資料中，即使飛機環繞地球赤道一圈，航段所橫跨的 cell 數也才 36 個，所以 worst case 是完全不會出現的。

表 4-5 檢查法比較表

cell 數	直接檢查法	MBR 檢查法	
	公式(2)	MBR 相交檢查	公式(2)
450( $12 \times 12$ )	1800 次	450 次	12 次
648( $10 \times 10$ )	2592 次	648 次	12 次
800( $9 \times 9$ )	3200 次	800 次	12 次

### 4.3 link 和 turn 介紹與說明

圖 4-5 為經過『GetMajorVector 模組』處理後的平均線段，對照圖 4-1 的原始航線圖，可看到 cell[28, 11]和 cell[29, 11]平均線段中斷或位移的情況，而非如 cell[30, 11]與 cell[29, 11]是正常相連的。我們以圖 4-7 分析 cell 間平均線段的連接問題。在圖 4-7(a)中，cell[0]和 cell[1]的平均線段雖然中斷，但若取兩個點的中點作為連接點，平均線段會與實際移動狀態對應(如大箭頭)。圖 4-7(b)中我們可以看到 cell[0]和 cell[1]的平均線段中斷，cell[4]和 cell[5]也是，若採用同上述連接方式，可發現所得到的結果(圖 4-7(b)的兩個大箭頭)，無法表示兩條從 cell[5]開始，經過 cell[4]最後走到 cell[0]的航線。換句話說，我們並不希望將 cell[0]與 cell[1]之間的平均線段相連。由於我們希望一條相連的平均線段，可以表示實際航段移動狀態，所以我們記錄航段在 cell 間的實際移動(link)行為，以便『Merge 模組』連接平均線段。

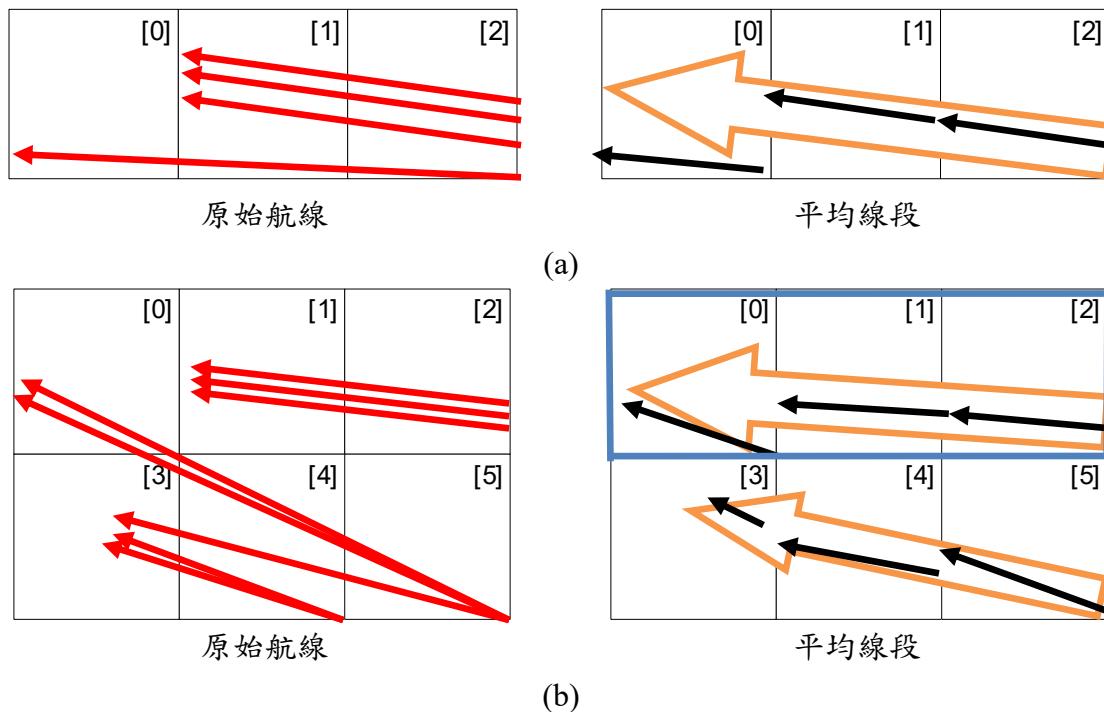


圖 4-7 cell 間平均線段連接問題

為了解決上述問題，我們使用了 link 矩陣，其目的在於記錄 cell 間航段移動情況，並統計 cell 中每個方向有多少航段移動，表 4-6 為矩陣結構，註標  $m$ 、 $n$  對應 cell 編號， $d$ (direction) 表示航段方向， $p$ (position) 表示從  $\text{cell}[m, n]$  移動到的 cell 方位代號，如下圖 4-8 往左上方( $d$  為 8)移動的航段，可能移動的方位為 1、7、8，元素 count 則記錄該移動方式有幾條航段通過。

表 4-6 link 矩陣結構

link					
	註標				元素
欄位名	$m$	$n$	$d$	$p$	count
說明	對應 cell 座標 $m$	對應 cell 座標 $n$	航段方向	移動方位 代號	航段總數

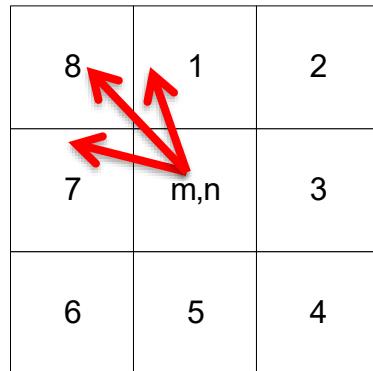


圖 4-8 移動方位

接著我們說明平均線段在 cell 內的問題，由於航線是由複數航段組成，所以會在 cell 中出現航線中轉的情況(航段連接航段)。如圖 4-9(a)的  $\text{cell}[0]$  中，實際的中轉點在下方，而上方則為航線交錯，但平均之後只形成一個交錯點。若是我们一律將此類的點視作中轉點，則有可能誤判。如圖 4-9(b)所示，在  $\text{cell}[0]$  中並無中轉點，平均後一樣形成一個交錯點，和圖 4-9(a)右圖的結果圖無法分別。我們希望平均線段的相連狀態可以表示實際移動狀態，也就是說我們必須知道航線在 cell 內是否中轉，才能將實際有發生中轉情況的平均線段連接起來(如圖 4-9(a)的大箭頭)，所以需要將航線在 cell 內的情況記錄下來(turn)，以便『Merge 模組』連接 cell 內的平均線段。

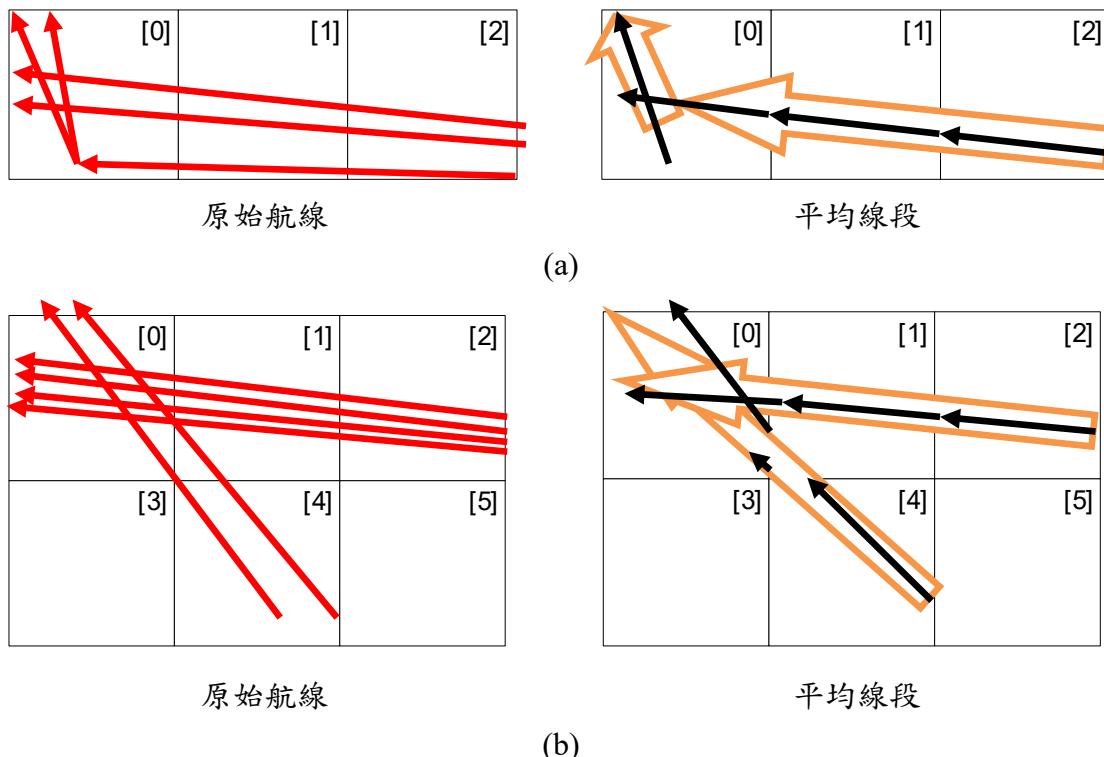


圖 4-9 cell 內平均線段中轉問題

表 4-7 為 turn 的矩陣結構，目的在於記錄單一 cell 內航線中的航段改變方向的模式，如圖 4-10，turn 矩陣註標  $m$ 、 $n$  對應 cell 編號， $d_1$  為進入 cell 的航段方向， $d_2$  為離開 cell 的航段方向，元素 count 為符合該移動方向的航段總數。

表 4-7 turn 矩陣結構

turn					
註標					元素
欄位名	$m$	$n$	$d_1$	$d_2$	count
說明	對應 cell 座標 m	對應 cell 座標 n	進入 cell 的 航段方向	離開 cell 的 航段方向	航段總數

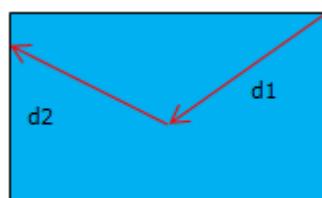


圖 4-10 turn 轉折範例

接著我們介紹『GetMajorVector 模組』所呼叫的 madeline 演算法。該演算法為了輸出正確的航段移動過程於 link 結構中，必須將記錄在 Set 中的 cell 編號排序，這是由於在 GetMajorVector 模組中 route 矩陣與 cell 矩陣是用迴圈依序檢查後，將有相交的 cell 編號記錄至 Set 中。所以如圖 4-11 所示，記錄在 Set 中的 cell 編號順序為 $(28, 12) \rightarrow (28, 13) \rightarrow (29, 11) \rightarrow (29, 12) \rightarrow (30, 11)$ ，但航段通過 cell 的實際順序為 $(30, 11) \rightarrow (29, 11) \rightarrow (29, 12) \rightarrow (28, 12) \rightarrow (28, 13)$ ，所以必須額外將航段排序(以起點開頭，迄點結尾)。排序想法如表 4-8，我們以圖 4-11 做說明範例，此範例航段的  $d$  為 8，所以 cell 經度編號  $m$ ，大小順序應為由大至小，並在  $m$  相同的情況，接著排序 cell 緯度編號  $n$ ，大小順序為由小至大。

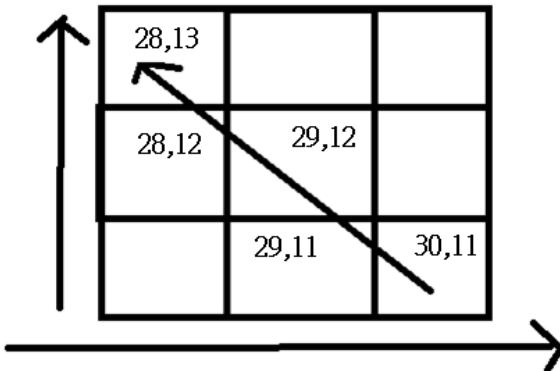


圖 4-11 航段經過 cell 順序範例圖

表 4-8 排序說明

$d$	方向性	經度編號 $m$ 順序	緯度編號 $n$ 順序
1	上	無關係	由小至大
2	右上	由小至大	由小至大
3	右	由小至大	無關係
4	右下	由小至大	由大至小
5	下	無關係	由大至小
6	左下	由大至小	由大至小
7	左	由大至小	無關係
8	左上	由大至小	由小至大

圖 4-12 為 madeline 演算法，在 L01 到 L04，使用迴圈將 Set 中的 cell 編號，依序儲存至 swapcell 二維矩陣中，矩陣內容說明如表 4-9，sortnum 則為 Set 中的 cell 個數(L05)。在 L06 到 L17，依照航段方向的不同將 swapcell 矩陣排序，如表 4-8，我們使用 sort 函式進行排序，輸入 swapcell 矩陣與其矩陣大小 sortnum，然後先排序經度編號，再排序緯度編號，且 "b" 代表由大排至小，"s" 代表由小排至大。根據排好序的 swapcell 陣列存入 link 矩陣中，於 L19 中使用 checkp 函數，找出下一個 cell 位於前一個 cell 的哪個方位。接下來，在 L22 到 L26 中，先判

斷此次處理的航段是否不為第一段，且  $d$  與前一段的航段方向 front 不同，如為是再判斷其所在起點與前一個航段的迄點是否位於同一個 cell 中，若此航線是由複數的航段組成，當以上的條件成立時就代表航線在此 cell 中轉，所以在 L24 加總至 turn 矩陣中。最後在 L27 至 L30 將此航段最後移動到的 cell 編號記錄至 checkm、checkn，並將 front 改為此航段的方向  $d$  且將 swapcell 矩陣清空。請注意如果 L22 判斷時 front 為 0，則表示此次處理的為航線中的第一段航段，所以不可能發生中轉情況，而 checkm、checkn 也不會在還沒記錄到 cell 編號的情況下就進行判斷(L23)。

表 4-9 swapcell 矩陣結構

swapcell		
欄位名	序號	code
說明	cell 個數	cell 編號 m 記錄在 0， n 記錄在 1

演算法名稱：madeline

輸入： front：記錄前一個航段種類  $d$

checkm、checkn：記錄前一段航段最後移動到的 cell 編號

$d$ ：航段種類

Set：儲存與航段相交的 cell 集合

變數： swapcell 二維矩陣：儲存 Set 中的 cell 編號

sortnum：swapcell 矩陣的大小

$p$ ：航段移動方位代號

輸出： turn 矩陣

link 矩陣

front、checkm、checkn

```

L01 for each Set[i]
L02     swapcell[i, 0] ← Set[i].m
L03     swapcell[i, 1] ← Set[i].n
L04 end for
L05     sortnum ← |swapcell|
L06     if (  $d = 1 \parallel d = 8$  ) than
L07         sort(swapcell, sortnum, "b", "s") //經度由大至小，緯度由小至大
L08     end if
L09     if (  $d = 2 \parallel d = 3$  ) than
L10         sort(swapcell, sortnum, "s", "s") //經度由小至大，緯度由小至大
L11     end if
L12     if (  $d = 4 \parallel d = 5$  ) than

```

L13	sort(swapcell, sortnum, "s", "b"); //經度由小至大，緯度由大至小
L14	<b>end if</b>
L15	<b>if</b> ( $d = 6 \parallel d = 7$ ) <b>than</b>
L16	sort(swapcell, sortnum, "b", "b") //經度由大至小，緯度由大至小
L17	<b>end if</b>
L18	<b>for</b> $i \leftarrow 0$ <b>to</b> sortnum -2 <b>do</b>
L19	$p \leftarrow \text{checkp}(\text{swapcell}[i, 0], \text{swapcell}[i, 1], \text{swapcell}[i + 1, 0], \text{swapcell}[i + 1, 1])$
L20	link[swapcell[i, 0], swapcell[i, 1], d, p].count++
L21	<b>end for</b>
L22	<b>if</b> ( front != 0 & front != d )
L23	<b>if</b> ( checkm = swapcell[0, 0] & checkn = swapcell[0, 1] )
L24	turn[checkm, checkn, front, d].count++
L25	<b>end if</b>
L26	<b>end if</b>
L27	checkm $\leftarrow$ swapcell[sortnum, 0]
L28	checkn $\leftarrow$ swapcell[sortnum, 1]
L29	front $\leftarrow$ d
L30	swapcell $\leftarrow$ NULL

圖 4-12 madeline 演算法

表 4-10 為表 4-1 的範例經過 madeline 演算法後的結果，總共有四筆航線資料，詳細移動路徑可參照圖 4-4。我們以第一條航線作為演算法範例，首先第一段航段參數  $d$  為 6，front 為 0，checkm 與 checkn 為  $[-1, -1]$ ，Set 為  $\{[30, 11], [29, 11]\}$ 。在 L01 到 L04 中將 Set 內的 cell 編號存入 swapcell 矩陣，在 L05 中得到 swapcell 矩陣大小為 2，由於  $d$  為 6，所以在 L15 到 L17 中先排序經度(由大至小)，然後再排序緯度(由大至小)，最後 swapcell 矩陣內容為  $\{[30, 11], [29, 11]\}$ 。接著在 L18 到 L21 將排序完的矩陣內容轉換存入 link 矩陣中，在 L19 使用 checkp 函數檢查  $[29, 11]$  位於  $[30, 11]$  的哪個方位，然後得到方位  $p$  為 7，並在 L20 記錄至 link[30, 11, 6, 7].count，最後在 L27 到 L30 回傳 checkm 與 checkn 為  $[29, 11]$ ，front 為 6，並將 swapcell 清空。第二段航段參數  $d$  為 8，front 為 6，checkm 與 checkn 為  $[29, 11]$ ，Set 為  $\{[29, 11]\}$ ，在 L01 到 L04 中將 Set 內的 cell 編號存入 swapcell 矩陣，在 L05 中得到 swapcell 矩陣大小為 1，由於  $d$  為 8，所以在 L15 到 L17 中先排序經度(由大至小)，然後在排序緯度(由小至大)，最後 swapcell 矩陣內容為  $\{[29, 11]\}$ 。接著由於矩陣太小(大小至少為 2)不執行 L18 到 L21，在 L22 發現此航段不為第一條，且此航段的方向  $d$  不等於前一航段的方向 front，然後在 L23 發現此航段起點所在的 cell 為  $[29, 11]$ ，與前一航段迄點所在的 cell (checkm, checkn) 相同，所以可以知道此航線在此 cell 發生中轉，在 L24 記錄至 turn[29, 11, 6, 8].count。

最後雖然一樣在 L27 到 L30 回傳 checkm、checkn、front 並將 swapcell 清空，但由於此航線已經處理完，所以 checkm、checkn 與 front 參數會在 GetMajorVector 模組設定為 [-1, -1] 與 0。其他航線以此類推，第二條航線  $d$  為 6 從 [30, 11] 移動至 [29, 11]，並且中轉成  $d$  為 8 從 [29, 11] 移動至 [28, 11]；第三條航線  $d$  為 6 從 [30, 11] 移動至 [29, 11]，然後再移動至 [28, 11]；第四條航線  $d$  為 6 從 [30, 11] 移動至 [29, 11]，並且中轉成  $d$  為 8 從 [29, 11] 移動至 [28, 11]。將上述資料整理後如表 4-10(a)，再依照各個 cell 移動的方式加總，可得表 4-10(b) 與 (c) 的結果。

表 4-10 link 與 turn 範例結果

(a)

編號	航線	航段	原 cell	$d$	移動到 cell	是否移動	是否中轉
1	1	1	[30, 11]	6	[29, 11]	是	否
2	1	2	[29, 11]	8	[29, 11]	否	是
3	2	1	[30, 11]	6	[29, 11]	是	否
4	2	2	[29, 11]	8	[28, 11]	是	是
5	3	1	[30, 11]	6	[28, 11]	是	否
6	3	2	[29, 11]	6	[28, 11]	是	否
7	4	1	[30, 11]	6	[29, 11]	是	否
8	4	2	[29, 11]	8	[28, 11]	是	是

(b)

link					
註標				元素	符合的結果 編號
$m$	$n$	$d$	$p$	count	
29	11	6	7	1	6
29	11	8	7	2	4、8
30	11	6	7	4	1、3、5、7

(c)

turn					
註標				元素	符合的結果 編號
$m$	$n$	$d1$	$d2$	count	
29	11	6	8	3	2、4、8

## 4.4 Merge 模組介紹與 resultroute 結構說明

接下來我們先介紹 resultroute 矩陣，其功能為儲存變動後的平均線段，下表 4-11 為 resultroute 的矩陣結構，註標中的  $m, n$  對應 cell 矩陣大小，編號  $d$ (direction) 則表示線段方向，共 8 種， $od$ (起迄)標示為線段的起點或迄點。而元素內記錄的  $x, y$  為線段端點座標，count 則記錄此線段為多少線段加總。請注意 resultroute 矩陣與 avgcell 矩陣結構(表 4-2)極為相似，但是為儲存為了連接而修改後的最終線段(座標)。

表 4-11 resultroute 矩陣說明表

resultroute							
	註標				元素		
欄位名	$m$	$n$	$d$	$od$	$x$	$y$	count
說明	cell 編號	cell 編號	線段方向	起迄點	線段經度	線段緯度	線段個數

在此先介紹用來連接平均線段的演算法，smooth 演算法與 connection 演算法。smooth 演算法是取兩個端點的中點作為連接點，以連接平均線段的迄點與被連接平均線段的起點，但如果其平均線段已經發生中轉情況(因為計算過中轉點而移動過座標)，則表示該平均線段所代表的航段中，有的在 cell 內中轉(turn)移動到其他 cell(link)，而我們希望保留該中轉點，所以另外使用 connection 演算法，直接額外連出一條線段，將其儲存至矩陣 additional，並在最後與 resultroute 矩陣一起輸出。以圖 4-13(a)為例，雖然 smooth 演算法與 connection 演算法的輸入相同，但 smooth 演算法是直接連接平均線段，如圖 4-13(b)，也就是計算對應的 avgcell 矩陣座標，將其平均後儲存至 resultroute 矩陣，然而 connection 演算法則是不做計算，直接額外連接平均線段，如圖 4-13(c)，將對應的航段的迄點和連接航段的起點視為一條連接平均線段，然後儲存至 additional 矩陣中(字串格式為  $m, n, d, od, x, y, count$ )。

但由於平均線段中所整合的航段可能會到達不同的 cell，所以我們定義連接的規則如下，在演算法中我們會依照其航段方向  $d$ ，讀取其對應方位的 link 矩陣的 count，我們以  $d$  為 6 當範例，則航段只可能移動到該 cell 的方位 5、6 與 7，以此類推其他航段方向，如圖 4-14，然後將其連接 cell 中的 count 最大值設為 Max，接著再重新檢查一次對應方位的 link 矩陣，選擇 link 矩陣中的 count 符合 Max 的 cell 進行連接，如果有複數 cell 符合此條件則同時連接。

count=3 8	count=3 1	2
count=1 7	m,n	3
6	5	4

(a) 平均線段

8	1	2
7	m,n	3
6	5	4

(b) smooth

8	1	2
7	m,n	3
6	5	4

(c) connection

圖 4-13 演算法範例

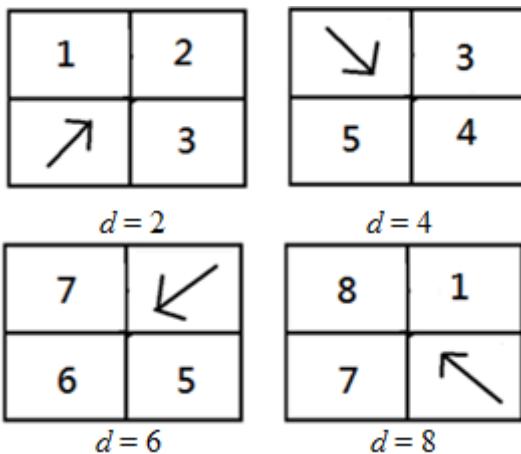


圖 4-14 航段方向對應的 cell 方位

圖 4-15 為『Merge 模組』，在程式 L01 先讀取 turn 矩陣，如果 turn 矩陣中有值，表示此 cell 內有中轉情況，需要將 cell 中的航段 d1 與航段 d2 相連接，如圖 4-10，所以在 L03 與 L04 中使用 hub 演算法計算中轉點，其規則為輸入 avgcell 矩陣中 d1 的迄點座標與 d2 的起點座標，然後計算其平均座標。例如我們從表

4-10(c)檢查 cell 狀況，發現 cell[29, 11]中有中轉情況，航段方向  $d$  從 6(往左下方移動)轉至 8(往左上方移動)，在參照表 4-4(b)得到 avgcell 線段座標後，計算 avgcell[29, 11, 6, 1]( 112.655, 22.0343725649351)與 avgcell[29, 11, 8, 0]( 113.54, 22.1601666666667)的平均座標為(113.0975, 22.0972696158009)，將其存入 resultroute[29, 11, 6, 1]與 resultroute[29, 11, 8, 0]中，如圖 4-16(b)的  $D$  點與  $E$  點會變成圖 4-16(c)的  $D'$ 點。接下來，在 L05 到 L16 以三維矩陣 type 記錄此 cell 中不同方向的航段是否已經計算過(type 中的值標示計算的程度，1 為迄點，2 為起點，3 表示起迄點都做過，0 則表示都沒做過)，接續上述範例，在此記錄 type[29, 11, 6]為 1，type[29, 11, 8]為 2。

然後在 L19 至 L41 讀取 link 矩陣，由於 link 矩陣只記錄 cell[m, n]移動到的方位代號  $p$ ，所以在 L21 中使用 pcheck 演算法得到實際移動到的 cell 編號，並記錄至 pm、pn 中。我們以表 4-10(b)的第一筆資料做為範例，使用 pcheck 演算法(航段方向為 6，cell 為[29, 11]，移動方位  $p$  為 7)，可知道該航段移動到 cell[28, 11]。請注意接下來所使用的演算法，其輸入的 avgcell 矩陣註標值對應於 link 矩陣，且可以從輸入的 avgcell 矩陣註標中得知 cell 編號[m, n]、航段方向  $d$  與平均線段座標。接著在 L22 到 L27 檢查 type 矩陣，檢查此 cell 中的平均線段其起迄點是否都連接過中轉點，在 L22 檢查起點和 L25 檢查迄點，如果沒有就使用 extend 演算法將其延長至對應的 cell 邊界上，也就是計算該平均線段與 cell 邊界的交點，以便 smooth 演算法連接平均線段。我們以圖 4-16(b)編號為 1 的平均線段作為範例，輸入參數 1、avgcell[29,11,8,0]與 avgcell[29, 11, 8, 1]後，會回傳延長後的端點座標，在此第一個參數標示延長後的為起點(0)還是迄點(1)，由於 type 值為 2，所以可得知其起點已經做過中轉點計算，所以只延長迄點座標至 cell 邊界上，演算法參數需設為 1。請注意 extend 演算法一樣使用計算交點演算法，但是其中做出一些調整，原本的演算法中計算出來的交點需同時位於航段中與 cell 邊界上，但在此由於是要延長平均線段，所以計算出來的交點只需位於 cell 邊界上就可以了。將迄點 avgcell[29, 11, 8, 1] (111.043667, 25.3590068385547)與 cell[29, 11]邊界做相交計算，得到新的線段端點為(110, 25.339155128916)，如圖 4-16(b)的  $B$  點會延長成圖 4-16 (c)的  $B'$ 點。

由於先前曾使用過 hub 演算法連接過中轉點，我們不希望在連接平均線段時變動到相關的起迄點，所以在 L28 與 L29 中將檢查每個 cell 的 type，如果所連接平均線段的起點(type 為 0 或 2)與迄點(type 為 0 或 1)都沒有發生中轉狀態，就使用 smooth 演算法，如果發生就使用 connection 演算法。參照表 4-10(b)，從 link [8, 29, 11, 7]中得知，cell[29, 11]中航段以方向 8 移動至[28, 11](使用 pcheck 演算法)，且在 L28 與 L29 檢查 type 發現所連接的起點(type[28, 11, 8]為 0)與迄點(type[29, 11, 8]為 2)都沒有發生中轉狀態，所以在 L30 與 L31 使用 smooth 演算法根據先前延長的端點(110, 25.339155128916)與 avgcell[28, 11, 8, 0]的座標(110, 23.982310257832)，計算平均值為(110, 24.660732693374)作為最後的連接點座標，並儲存至 resultroute[29, 11, 8, 1]與 resultroute[28, 11, 8, 0]中，也就是由圖 4-16(c)

的  $B'$  點(延長的端點)與  $A$  點計算出圖 4-16(d)的  $A'$  點(連接點)，並將變動後的線段標示為 smooth。

模組名稱：merge

輸入：avgcell 矩陣：為平均線段

link 矩陣：每條航段走的 cell 編號序列

turn 矩陣：不同  $d$  的航段如有相連，其連接點的序列

變數：type：判斷 cell 中是否中轉的變數，以三維矩陣記錄對應的 cell 編號

additional：額外線段，一維矩陣

$i$ ：additional 矩陣的計數器，起始為 0

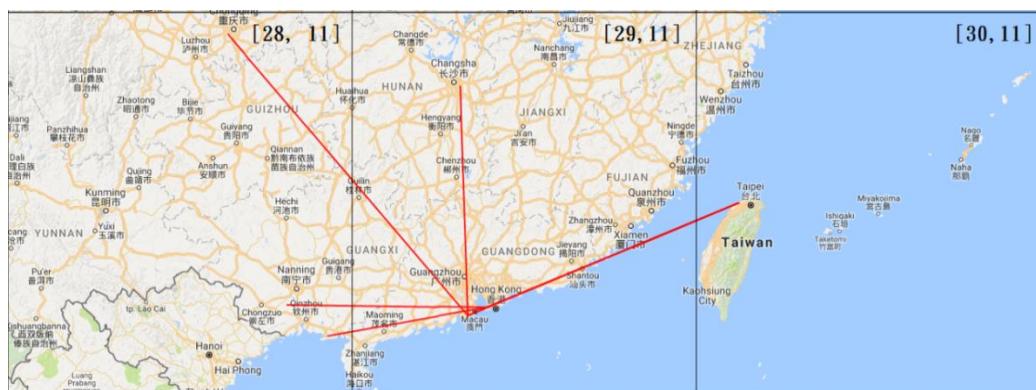
輸出：resultroute 矩陣與 additional 矩陣

L01	<b>for each</b> turn
L02	<b>if</b> (turn [ $m, n, d1, d2$ ].count > 0 )
L03	resultroute[ $m, n, d1, 1$ ] & resultroute[ $m, n, d2, 0$ ]
L04	$\leftarrow$ hub(avgcell[ $m, n, d1, 1$ ], avgcell[ $m, n, d2, 0$ ])
L05	<b>if</b> ( type[ $m, n, d1$ ] = 0 )
L06	type[ $m, n, d1$ ] $\leftarrow$ 1
L07	<b>end if</b>
L08	<b>if</b> ( type[ $m, n, d1$ ] = 2 )
L09	type[ $m, n, d1$ ] $\leftarrow$ 3
L10	<b>end if</b>
L11	<b>if</b> ( type[ $m, n, d2$ ] = 0 )
L12	type[ $m, n, d2$ ] $\leftarrow$ 2
L13	<b>end if</b>
L14	<b>if</b> ( type[ $m, n, d2$ ] = 1 )
L15	type[ $m, n, d2$ ] $\leftarrow$ 3
L16	<b>end if</b>
L17	<b>end if</b>
L18	<b>end for</b>
L19	<b>for each</b> link
L20	<b>if</b> ( link [ $d, m, n, p$ ].count > 0 )
L21	pm、pn $\leftarrow$ pcheck[ $m, n, p$ ]
L22	<b>if</b> ( type[ $m, n, d$ ] = 0    type[ $m, n, d$ ] = 1 )
L23	resultroute[ $m, n, d, 0$ ] $\leftarrow$ extend(0, avgcell[ $m, n, d, 0$ ], avgcell[ $m, n, d, 1$ ])
L24	<b>end if</b> //L22
L25	<b>if</b> ( type[ $m, n, d$ ] = 0    type[ $m, n, d$ ] = 2 )
L26	resultroute[ $m, n, d, 1$ ] $\leftarrow$ extend(1, avgcell[ $m, n, d, 0$ ], avgcell[ $m, n, d, 1$ ])
L27	<b>end if</b> //L25

L28	<b>if</b> ( type[ <i>m, n, d</i> ] = 0    type[ <i>m, n, d</i> ] = 2    type[pm, pn, <i>d</i> ] = 0
L29	type[pm, pn, <i>d</i> ] = 1 )
L30	additional [ <i>i</i> ] $\leftarrow$ connection(avgcell[ <i>m, n, d, 1</i> ], avgcell[pm, pn, <i>d, 0</i> ])
L31	<i>i</i> = <i>i</i> + 2
L32	<b>else</b>
L33	resultroute[ <i>m, n, d, 0</i> ] & resultroute[ <i>m, n, d, 1</i> ]
L34	$\leftarrow$ smooth ( avgcell[ <i>m, n, d, 1</i> ], avgcell[pm, pn, <i>d, 0</i> ] )
L35	<b>end if</b> //28
L36	<b>end if</b> //29
L37	<b>end for</b>

圖 4-15 merge 模組

最後請注意圖 4-16(b)中編號為 2 的平均線段，雖然也為中斷狀態，但因為在 L22 到 L27 檢查其中轉狀態時，發現 type[29, 11, 6]為 1，這表示其連接的平均線段迄點(avgcell [29, 11, 6, 1])發生中轉狀態，且雖然 type[28, 11, 6]為 0，但由於其起點也已經在 cell 邊界上，所以 extend 演算法也沒有變動其座標，也就是說由於圖 4-16(b)中編號 2 線段有中轉情況發生，並使用 hub 演算法變動過座標，所以 *D* 點不延長，維持如圖 4-16(c)的 *D'*點。接著在 L28 與 L29 發現 type[29, 11, 6]為 1，所以在 L30 和 L31 使用 connection 演算法連接一條新的線段，將 avgcell[29, 11, 6, 1]作為起點，avgcell[28, 11, 6, 0]作為迄點後，儲存至 additional[i] 中，其線段使用起點所在的 cell 作為編號，計數器 *i* 加 2(起始為 0)，也就是將圖 4-16(c)的 *D'*點與 C 點做連接，結果如圖 4-16(d)標示為 connection 的線段。完整的 resultroute 和 additional 矩陣內容如表 4-12 所示。



(a) Input

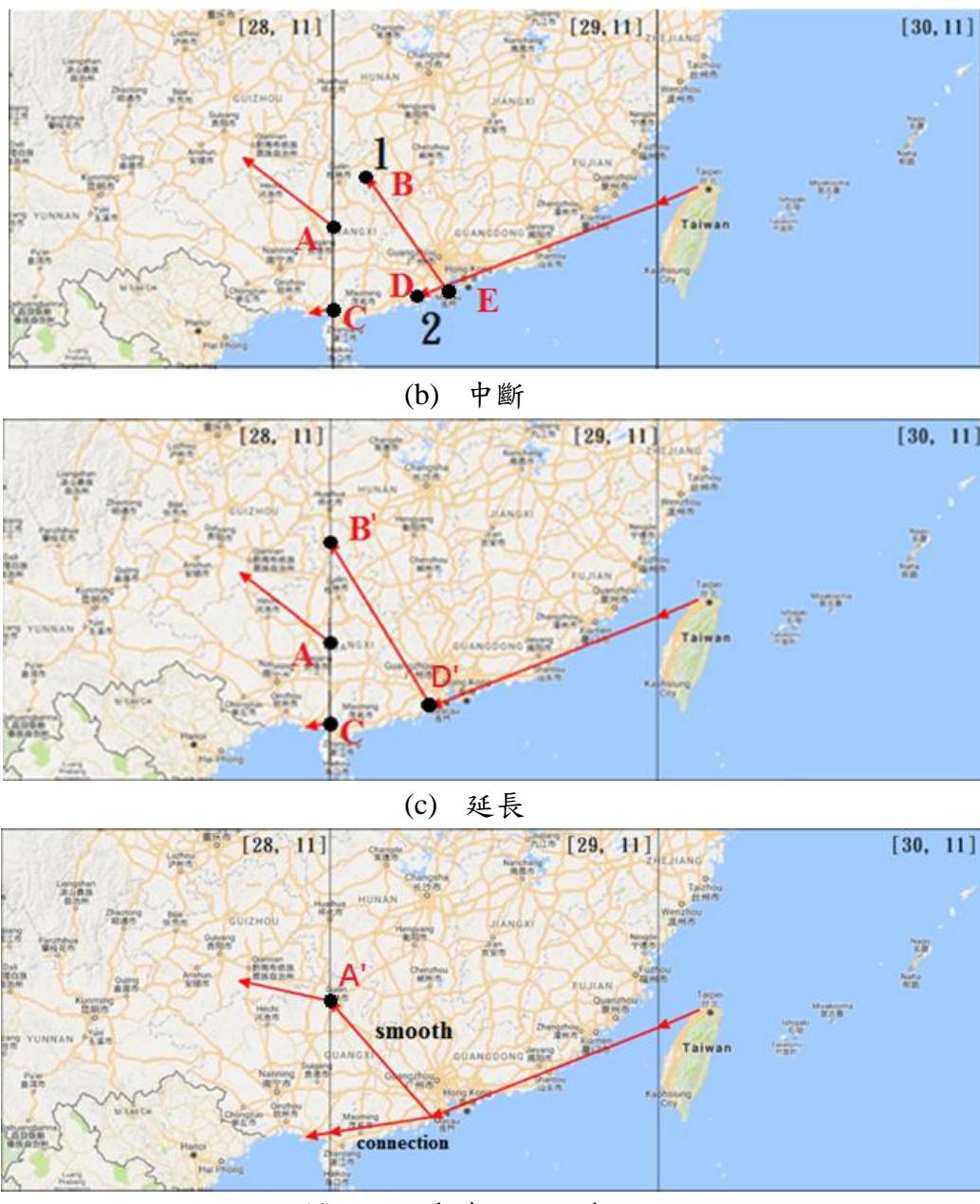


圖 4-16 延長(extend)方法

表 4-12 resultroute 矩陣與 additional 矩陣內容

resultroute					
註標				元素(平均後的值)	
<i>m</i>	<i>n</i>	<i>d</i>	od	<i>x</i>	<i>y</i>
28	11	6	0	110	21.6569902597403
			1	110	21.6569902597403
		8	0	110	25.339155128916
			1	107.242	25.8974

29	11	6	0	113.0975	22.0972696158009
			1	110	21.6569902597403
			8	0	113.0975
				1	110
30	11	6	0	121.224	25.0764
			1	120	24.6121718720177

additional						
<i>m</i>	<i>n</i>	<i>d</i>	od	<i>x</i>	<i>y</i>	
29	11	6	0	120	24.6121718720177	
			1	113.0975	22.0972696158009	

## 4.5 Display 模組與介紹

Display 模組如下圖 4-17 所示，將最後的摘要化路徑利用 Google Map 網頁顯示，請注意 Google Map 的 API 是使用 javascript 語法實現，詳細說明如下表 4-13。請注意由於摘要化路徑是由不同數量的航段所整合成的，而我們希望可以用顏色區隔其不同，所以此模組中設定了一個門檻值 *t* (目前設為 input 資料數的 1/10)。程式在 L01 讀取所有的摘要化路徑(resultroute 矩陣與 additional 矩陣)，然後在 L02 與 L03 使用紅線標示 count 值大於門檻值的摘要化路徑，在 L04 與 L05 使用綠線標示其他小於門檻值的摘要化路徑。根據表 4-12 的 resultroute 陣列與 additional 陣列，範例結果為圖 4-16(d)，在此門檻值為 0.2，所以路徑都為紅色。

表 4-13 javascript 語法說明

javascript 語法	說明
var map = new google.maps.Map (document.getElementById('map_canvas'),mapOptions)	定義頁面上的單個地圖
var mapOptions = { zoom: 2, center: new google.maps.LatLng(y,x)}	地圖的詳細設定
var Path = new google.maps.LatLng(y,x),new google.maps.LatLng(y,x), new google.maps.LatLng(y,x),new google.maps.LatLng(y,x)	路徑(線段)所經過的座標
var flightPath = new google.maps.Polyline({path: Path, geodesic:false, strokeColor:'black', strokeOpacity:1.0, strokeWeight:1,map: map})	使用普通線段顯示(cell 所用)

var lineSymbol = { path: google.maps.SymbolPath.FORWARD_CLOSED_ARROW}	IconSequence.offset 屬性，將箭號新增到線段的終點。
var flightPath = new google.maps.Polyline({path: Path, geodesic:false, icons: [{icon: lineSymbol,offset:'100%'}], strokeColor:'red', strokeOpacity:1.0, strokeWeight:1, map: map})	使用箭頭線段顯示(摘要化路徑所用)
細部說明	
zoom：設定地圖的初始解析度，其中縮放 0 對應完全縮小的地球地圖，較高的縮放層級則會放大到較高的解析度。	
center：地圖的中心座標	
geodesic：為 turn 時線段為曲線，為 false 時線段為直線	
strokeColor：線段的筆觸色彩。支援所有 CSS3 色彩，延伸的具名色彩除外	
strokeOpacity：線段的筆觸不透明度介於 0.0 和 1.0 之間	
strokeWeight：線段的筆觸寬度 (以像素為單位)	
map：表示線段顯示於地圖中	
icon：在線段上所使用的符號。	

模組名稱：Display 模組

輸入：additional 矩陣

resultroute 矩陣

t：門檻值

輸出： Google Map

L01	<b>for each line in ( resultroute &amp; additional)</b>
L02	<b>if (line.count &gt; t )</b>
L03	draw line with red
L04	<b>else</b>
L05	draw line with green
L06	<b>end if</b>
L07	<b>end for</b>

圖 4-17 Display 模組

## 第 5 章 系統呈現與實驗結果

在本章中，我們說明本系統輸出的摘要化路徑，並進行實驗來分析所提出方法在不同的輸入下效率上的差異。首先說明我們進行實驗的環境，我們以個人電腦作為實驗的環境，其 CPU 為 Intel Core i7-2600 四核心，核心時脈為 3.4GHz，而記憶體為 4GB，所採用的作業系統為 64 位元的 Windows 10 企業版。資料前置處理階段中的資料庫建構於 SQL Server 2012，而系統架構則以 Microsoft Visual C++ 2010 進行實作，最後將摘要化路徑呈現在 Google Maps 上。

### 5.1 資料集說明

本論文所研究的旅客旅行路徑資料總共有 20731 筆，主要為 2013 與 2014 年以台灣為起點或迄點的旅行路徑。由於原始路徑方向交錯難以進行摘要化，所以我們使用 SQL 查詢抓取特定方向的資料，得到不同的資料集。例如圖 5-1 為台北至歐洲的 2013 年旅客旅行路徑資料，其中的點代表機場位置，線代表兩機場間存在航段，對應的 SQL 如表 2-7 所示。

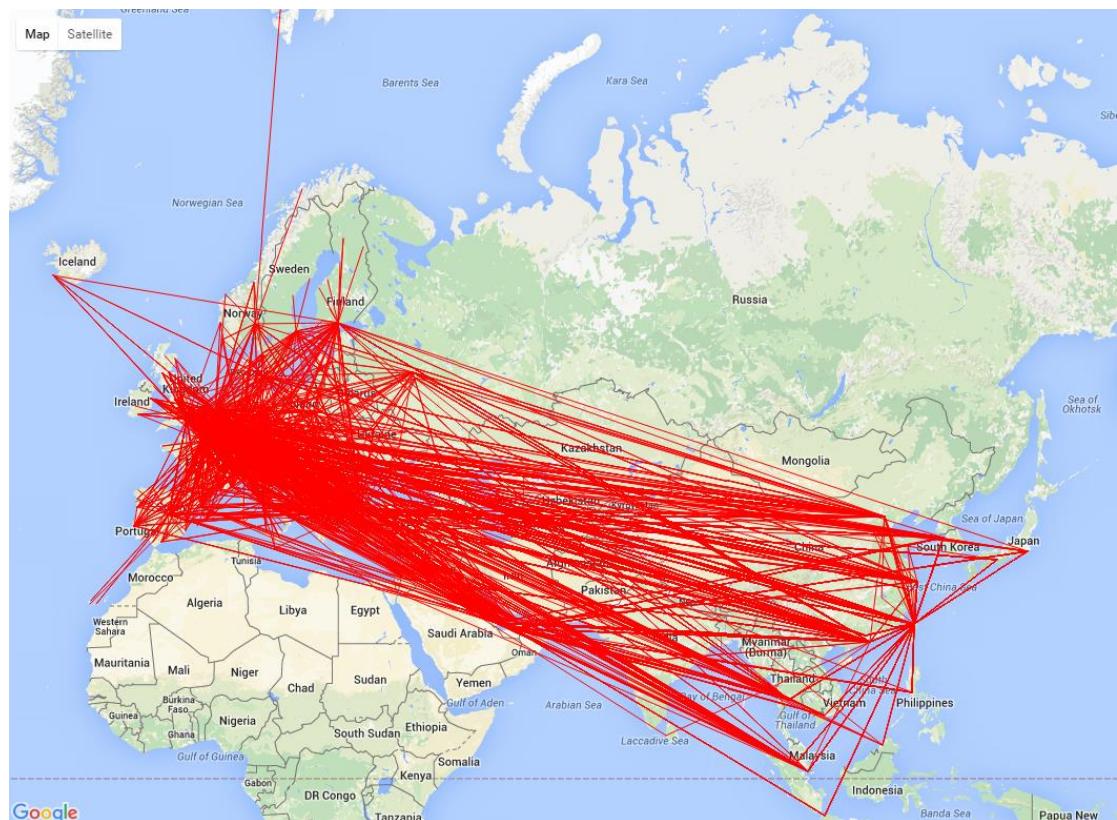


圖 5-1 2013 年台北至歐洲的原始旅客旅行路徑圖

我們在表 5-1 中列出本系統所使用的資料集，為 2013 年旅客數不為 0 的旅客旅行路徑資料，分別為以台灣為起點，然後抵達非洲、亞洲與大洋洲、歐洲、美洲等四大區域的資料集。

表 5-1 不同資料集

起點	迄點	資料筆數
台灣	非洲	231 筆
	亞洲與大洋洲	1923 筆
	歐洲	2179 筆
	美洲	2638 筆

## 5.2 系統呈現說明與討論

我們將本系統處理 4 個資料集所得到的旅客旅行路徑圖，依序講解與分析。實驗環境是依照地理位置將地球切成  $10 \times 10$  的 cell，cell 總共有 648 格，表 5-2 總計摘要化後的路徑數變化。

表 5-2 路徑筆數的變化

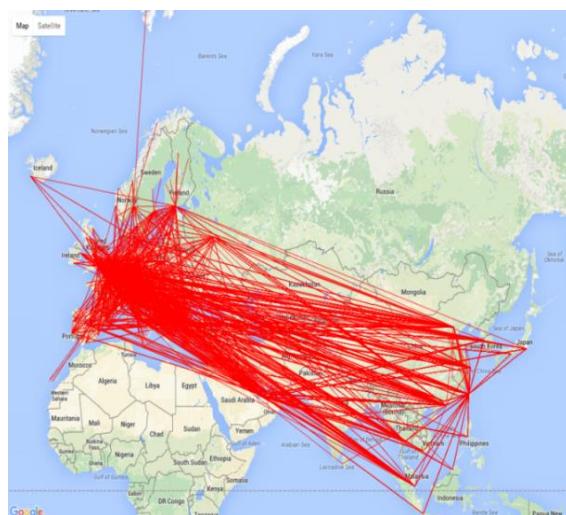
資料集	原始旅行路徑	摘要化路徑
非洲	231 筆	179 筆
亞洲與大洋洲	1923 筆	741 筆
歐洲	2179 筆	776 筆
美洲	2638 筆	1104 筆

首先，圖 5-2 為台灣至歐洲的旅行路徑圖，其中圖 5-2(a)為原始旅客旅行路徑資料，圖 5-2(b)為摘要化的旅客旅行路徑圖。雖然摘要化後路線由原來的 2179 筆，簡化為 776 筆，可較明確顯示路徑移動與方向，但無法分辨出最熱門的路徑。所以我們將旅行路徑其對應總筆數超過總數  $1/10$ (也就是 217 筆)的路徑，以不同顏色標示，以便讓使用者知道哪些路徑為多數航線移動的方式，並在圖 5-2(c)單獨顯示。由該圖的結果可知道到達歐洲的熱門路徑主要有三種，分別為以中國港澳、日韓與東南亞中轉，且大部分的航線移動到歐洲後，以德國與法國交界處作為中轉區，移動到其他歐洲地區。

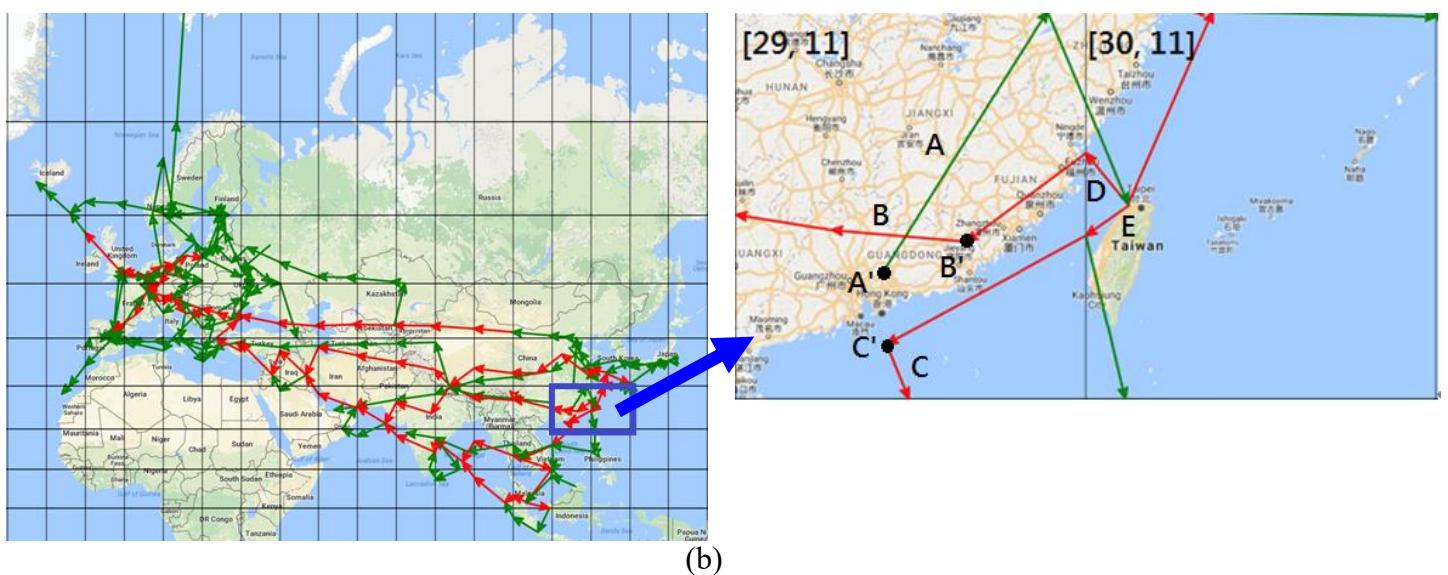
然後在觀察旅客旅行路徑圖後，我們發現了一些問題並依序說明。第一個問題是一個中轉機場會產生多個中轉點，此問題會發生在不同方向的路徑在相同 cell 中轉。這是因為本論文的方法是在 cell 內轉換方向時紀錄中轉，所以每個方向的中轉點都是分開記錄的。如圖 5-2(b)的 cell[29, 11]中，存在三條中轉路徑 A、B 與 C，由於其方向不同，所以中轉點分別為 A'、B'與 C'。接著在連接路徑時，

當連接最多航線經過的中轉點後，沒被連接到的中轉點其對應路徑就會產生中斷現象，如圖 5-2(b)路徑  $D$  與  $E$  分別連接路徑  $B$  與  $C$ ，而對應  $A'$  的路徑  $A$  就發生了中斷。然後我們還發現計算出的中轉點座標都偏移真實位置，這是因為合併路徑時是依照航段方向，並沒有細分此航段是否從此 cell 中轉，這就導致中轉點的位置被其他不在此 cell 中轉的航段影響到。

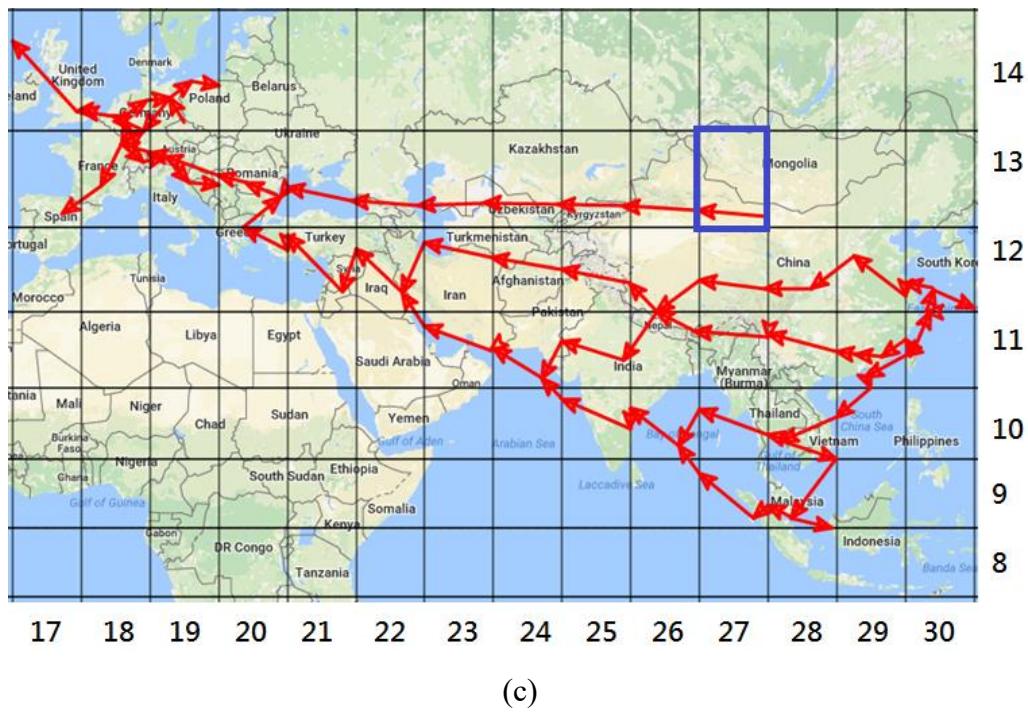
接著，在圖 5-2(c)也觀察到路徑中斷的現象，這是因為有些航線是當路徑移動到一半後才合併進來，讓路徑的中後半段剛好可以超過門檻值。比如位於蒙古下方的路徑(cell[27, 13])，原本從北京往左移動的摘要化路徑其航線數並沒有超過門檻值，但由於有些散亂航線是從中國內地移動至烏魯木齊，讓後半段路徑所對應之航線數約略超過門檻值，才造成一條起點並非在台灣的路徑。



(a)



(b)

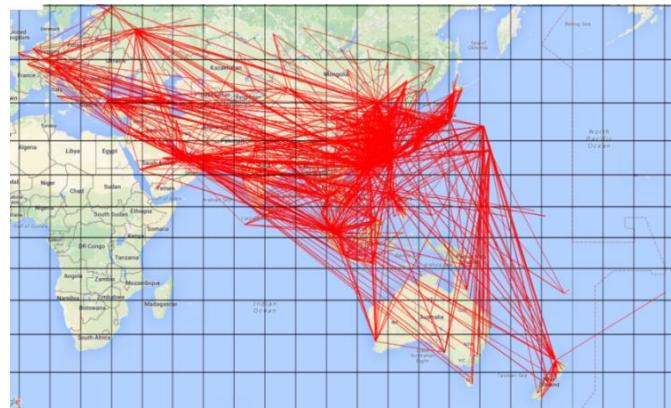


(c)

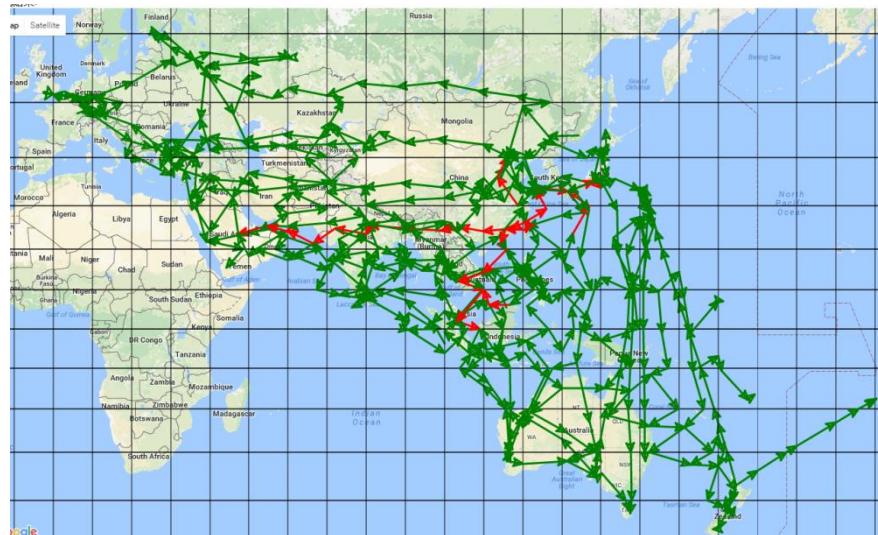
圖 5-2 台灣至歐洲的旅行路徑圖

其次，圖 5-3 為台灣至亞洲與大洋洲的旅行路徑圖，其中圖 5-3 (a)為原始旅客旅行路徑資料，圖 5-3(b)為摘要化旅客旅行路徑圖，圖 5-3(c)為對應航線最多的摘要化路徑(門檻值為 192 筆)。由結果可知雖然全部航線所移動到的範圍極大，但大多數的航線為移動到日本、韓國、中國或東南亞，此外還有一條往沙烏地阿拉伯的路徑。

請注意，圖 5-3(c)中台灣右側的 cell[31, 11]有一條多出的路徑 A，其原因跟先前提過圖 5-2 (c)中 cell[27, 13]路徑中斷的問題一樣，也就是當路徑移動到一半後有其他的航線合併進來。例如這個範例是因為航線從東南亞移動到日本時，當快移動到目的地的時候，分散的航線彙整起來使合併的航線數超過門檻值。其他如 cell[29, 12]的路徑 B 也是同樣的原因，為合併了許多不同方向但目的地為北京的航線。但 cell[29, 9]的路徑 C，則是抵達汶萊的航線數很少，但離開汶萊的航線數很多，所以造成了合併後的路徑航線數超過門檻值。所以我們得到一個結論，當摘要化路徑越接近中轉機場的時候，其路徑合併的航線數變化就會越大，而造成只有靠近中轉機場的摘要化路徑其航線數超過門檻值，並進一步形成中斷的現象。



(a)



(b)



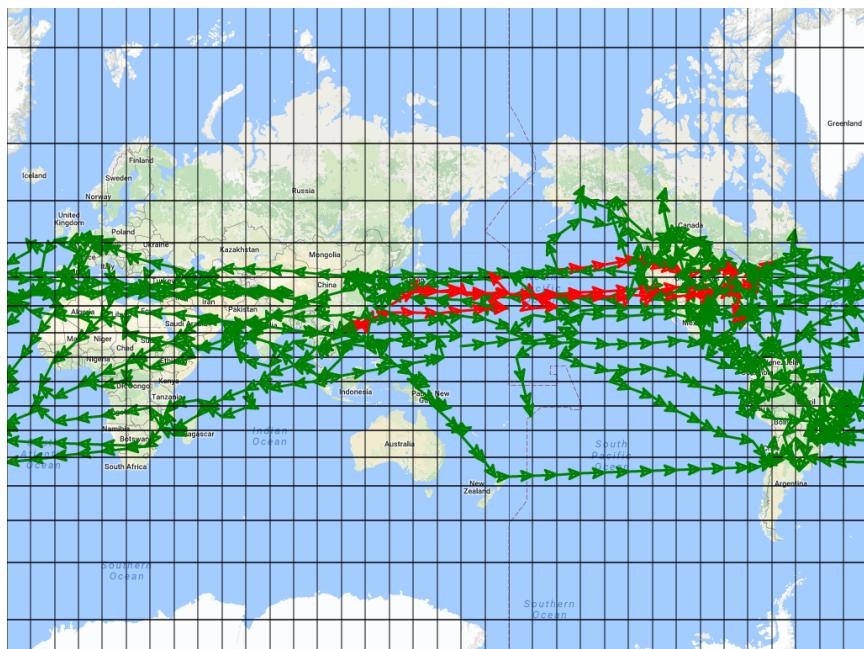
(c)

圖 5-3 臺灣至亞洲與大洋洲的旅行路徑圖

接著，圖 5-4 為台灣至美洲的旅行路徑圖，其中圖 5-4(a)為原始旅客旅行路徑資料，圖 5-4(b)為摘要化旅客旅行路徑圖，圖 5-4(c)為經過航線最多的摘要化路徑(門檻值為 263 筆)。由結果可知雖然航線所覆蓋的範圍極大，但大部分的航線還是以橫跨太平洋直接到美國為主，並分為直達或從日本中轉。



(a)



(b)

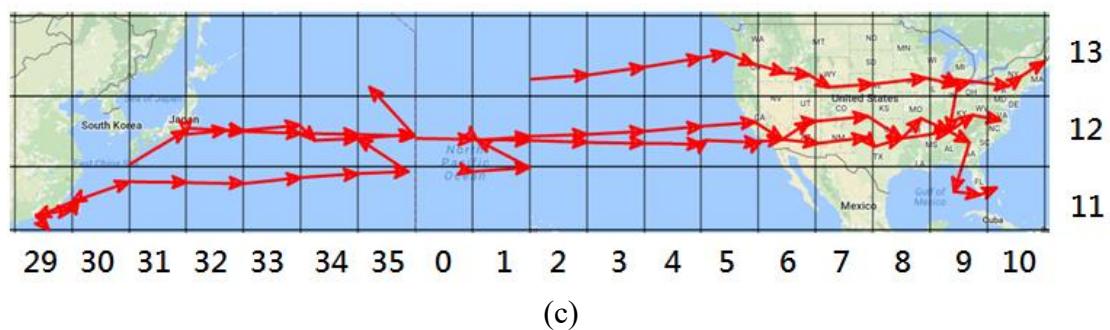
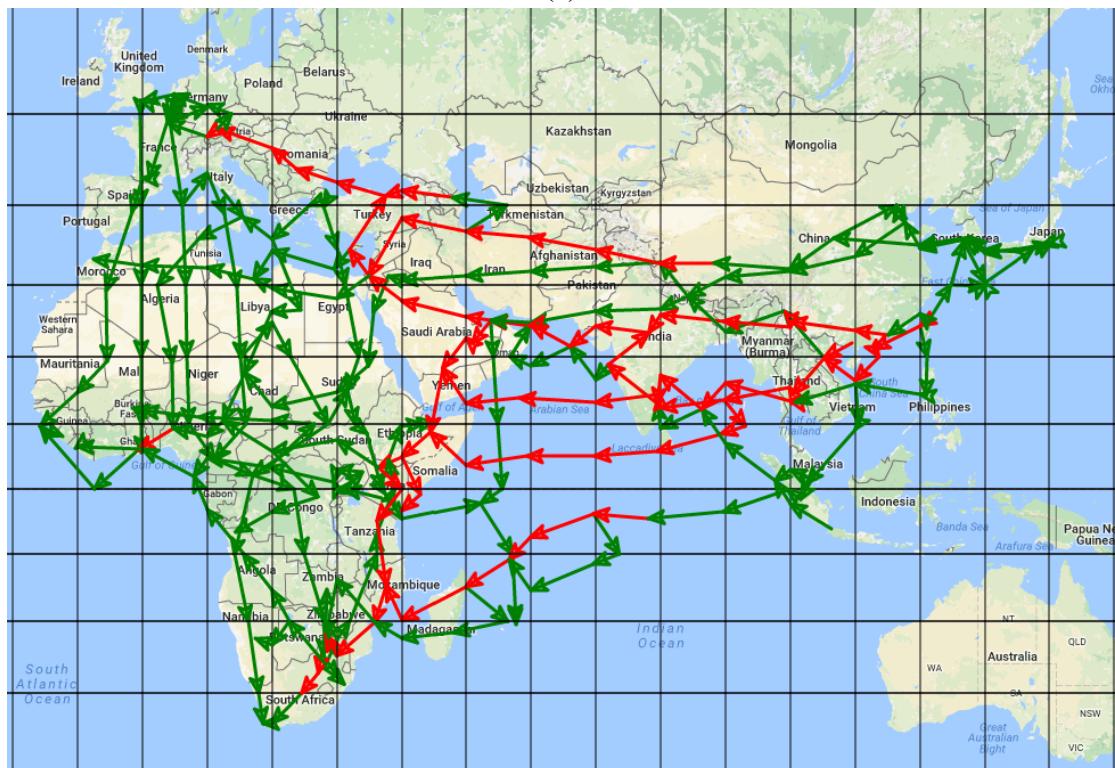


圖 5-4 台灣至美洲的旅行路徑圖

最後，圖 5-5 為台灣至非洲的旅行路徑圖，其中圖 5-5(a)為原始旅客旅行路徑資料，圖 5-5(b)為摘要化旅客旅行路徑圖，圖 5-5(c)為經過航線最多的摘要化路徑。請注意，由於非洲的旅行路徑移動方式過於平均(門檻值 23 筆)，所以參考性較低，但我們還是可以知道，大部分的航線移動路徑分為兩種，一種先移動到東非後，再一路沿海移動到南非；另一種則先移動到西亞後，再移動到南歐。請注意在此有一條獨立路徑 A，此路徑為奈及利亞移動至象牙海岸，其合併航線有 26 筆，約略超過門檻值一些，然後在比對圖 5-5 (a)後，發現此路徑合併了從西亞、東非與南非中轉的航線。



(a)



(b)



圖 5-5 台灣至非洲的旅行路徑圖

### 5.3 cell 大小對輸出呈現的影響

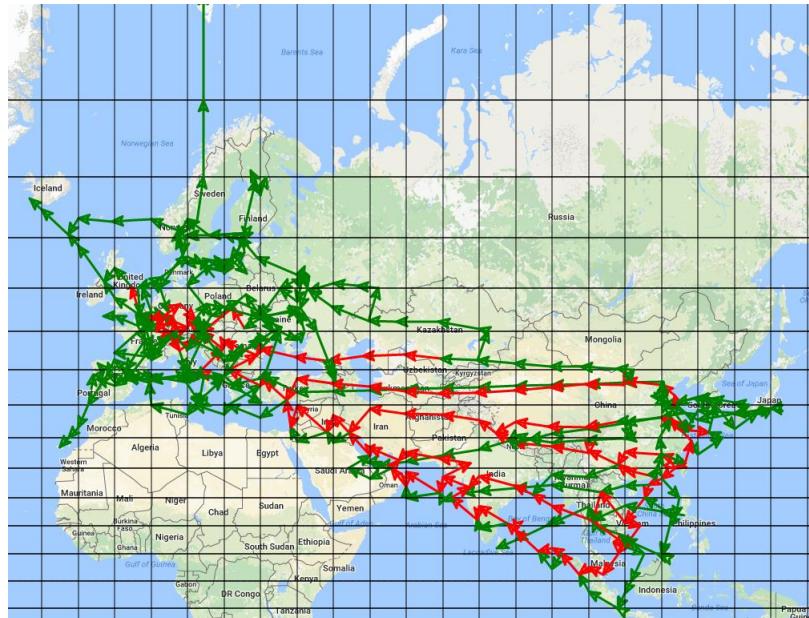
本節以台灣至歐洲的旅客旅行路徑為代表，分析 cell 大小對輸出呈現的影響，其中圖 5-6 (a) 的 cell 大小為  $15 \times 15$ ，圖 5-6 (b) 的 cell 大小為  $10 \times 10$ ，圖 5-6 (c) 的 cell 大小為  $8 \times 6$ 。我們仔細觀察這三張圖後可以發現，cell 如果切的越小，那摘要化路徑就越詳細，但資訊量會很多並且難以觀察；反之 cell 切的越大，則摘要化路徑就越簡略，只可以簡單了解其資料意義。所以，本系統採用中間的大小，也就是  $10 \times 10$  來進行實驗和呈現。



(a)  $15 \times 15$ (288 格)



(b)  $10 \times 10$ (648 格)



(c)  $8 \times 6$ (1350 格)

圖 5-6 cell 大小對系統呈現的影響實驗圖

## 5.4 cell 數量大小之實驗與效率評估

接下來的實驗，我們將資料集固定為歐洲(2179 筆路徑)，然後改變 cell 區域的大小，將其切割為不同格數(cell 越小，格數越多)，並測試在不同模組中所需要的時間，以下時間單位為毫秒。參考表 5-3 與圖 5-7，我們依序分析不同模組的耗時與原因：在 BuildRoute 模組中，耗時與 cell 格數無任何關係；在 BuildCell 模組中，其功能為切割 cell 範圍，但由於耗時極小，即便改變了 cell 大小，其耗時只有些微上升；在 GetMajorVector 模組中，其功用為計算航線與 cell 的交點，所以耗時與 cell 格數成正比，格數越多耗時就會越大；在 Merge 模組中，其目的為連結平均線段，由實驗數據觀察其耗時與 cell 格數似乎相關；最後在 Display 模組中，其目的為顯示摘要化路徑，耗時與 cell 格數約略相關。

表 5-3 不同 cell 大小之各模組耗時(ms)

cell 格數	BulidRoute	BulidCell	GetMajorVector	Merge	Display
162 格 (20 × 20)	5.8536	0.0069	109.3014	2.2286	10.5044
288 格(15 × 15)	5.8769	0.0087	186.9494	2.298	35.3448
648 格(10 × 10)	5.917	0.0601	355.8221	3.5169	67.0958
800 格(9 × 9)	5.2378	0.0441	413.0253	4.7105	105.6561
1350 格(8 × 6)	5.8953	0.0344	644.4185	7.2036	221.1655

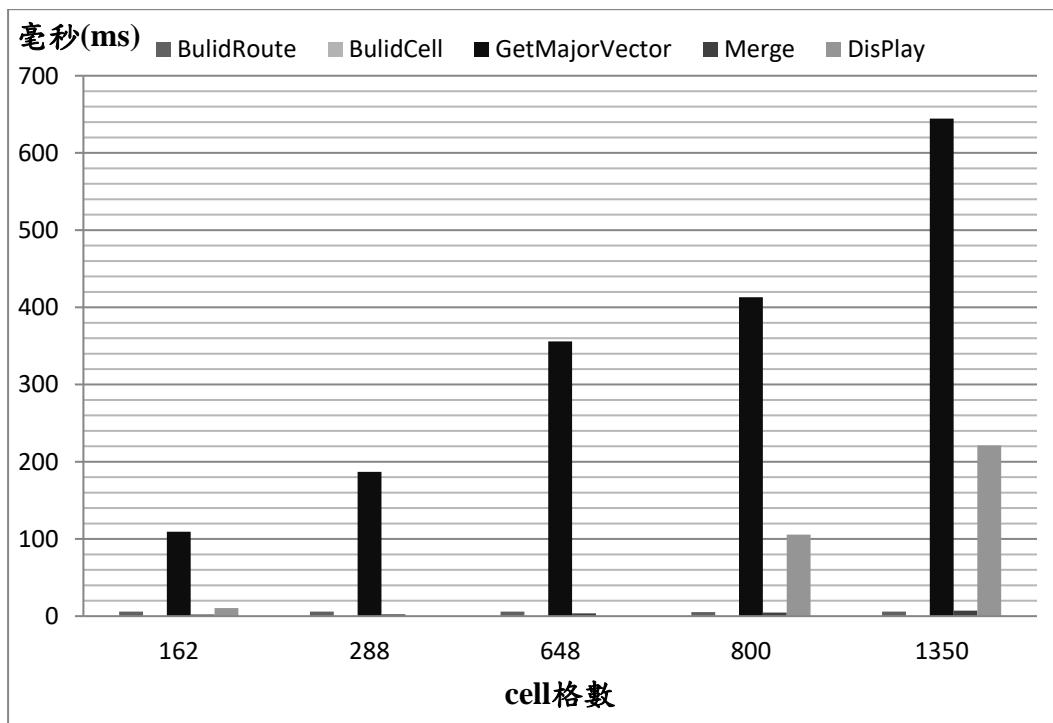


圖 5-7 cell 格數影響耗時之比較圖

由於 Merge 模組的輸入為矩陣 avgcell、link 與 turn，並非原始資料，所以我們對其做進一步的分析。從表 5-4，我們發現此三個矩陣的大小都與 cell 格數相關，也就是說如果 cell 格數越多，在 GetMajorVector 模組中輸出的矩陣也會越大，這也導致了 Merge 模組耗時的上升。

表 5-4 歐洲資料集在不同格數所輸出之矩陣大小

cell 格數	avgcell	link	turn	Merge
162 格 (20×20)	134	130	56	2.2286(ms)
288 格(15×15)	192	191	69	2.298(ms)
648 格(10×10)	326	325	78	3.5169(ms)
800 格(9×9)	388	375	79	4.7105(ms)
1350 格(8×6)	562	538	85	7.2036(ms)

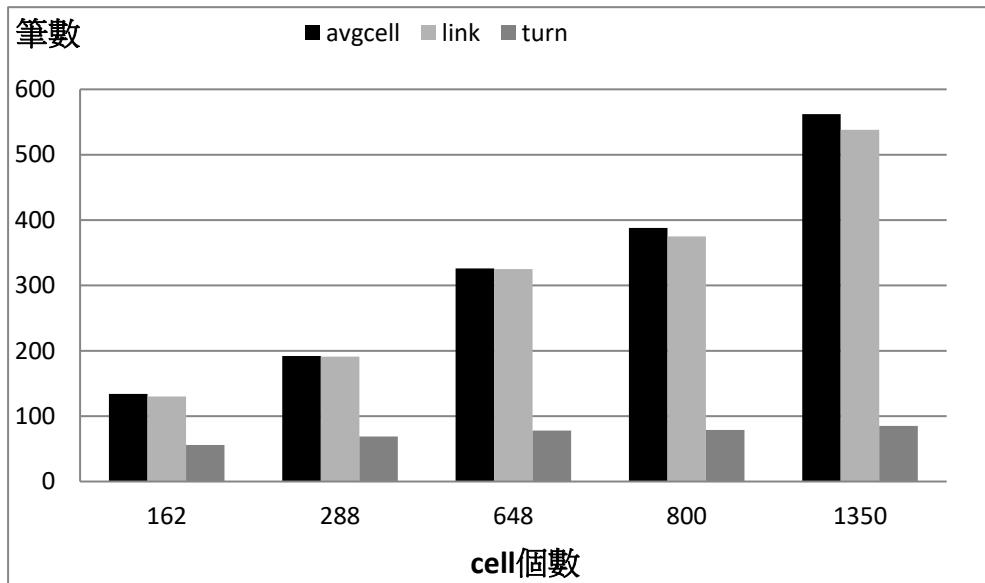


圖 5-8 不同格數大小的輸出矩陣比較圖

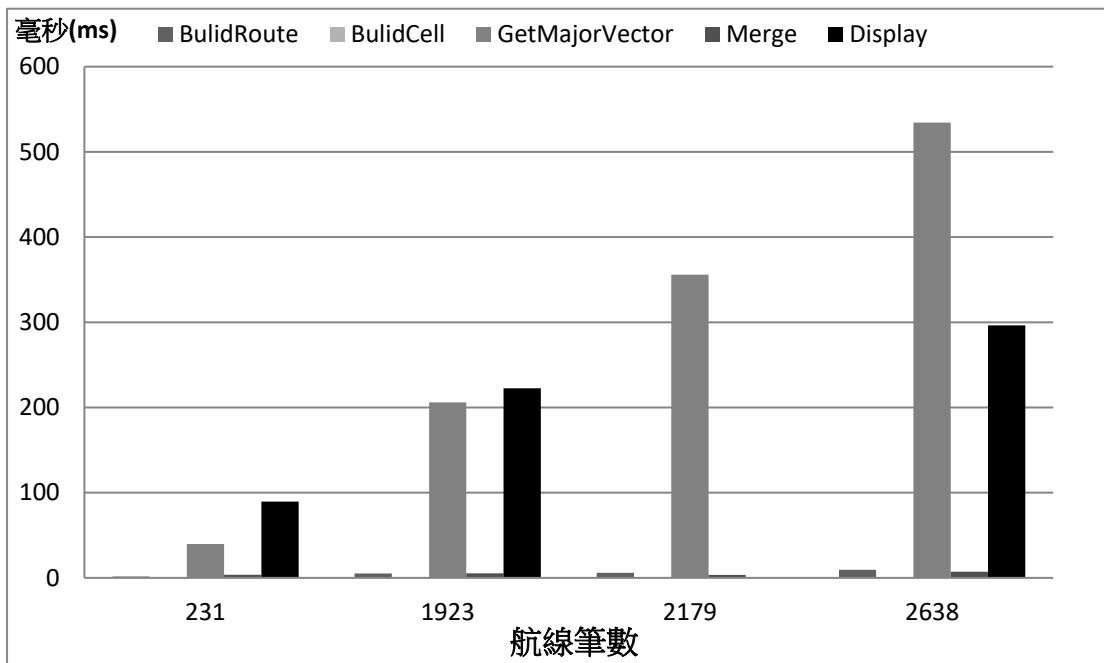
## 5.5 航線數量之效率評估

接下來的實驗，我們先測試表 5-1 四個不同資料集在相同的 cell 區域 ( $10 \times 10$ ) 下，系統中不同模組所需要的時間，以下時間單位為毫秒。參考表 5-5 與圖 5-9，我們依序分析不同模組的耗時與原因：在 BuildRoute 模組中，耗時會隨著航線數量而增加；在 BuildCell 模組中，耗時與航線數量無任何關係；在 GetMajorVector 模組中，耗時會與航線數量成正比；在 Merge 模組中，耗時與航線數量看起來無相關性；最後在 Display 模組中，耗時與航線數量無相關。

表 5-5 不同資料集各模組的耗時(ms)

毫秒(ms)	BulidCell	BulidRoute	GetMajorVector	Merge	Display
非洲(231 筆)	0.0347	1.6074	39.6361	3.6027	89.5775
亞洲與大洋洲(1923 筆)	0.0549	5.2291	206.0019	5.4017	222.5556
歐洲(2179 筆)	0.0601	5.917	355.8221	3.5169	67.0958
美洲(2638 筆)	0.0175	9.453	534.4105	7.1474	296.2773

圖 5-9 不同資料集之耗時比較圖



接著，我們固定資料集為從台灣到美洲(2638 筆)，並將資料大小細分為百分比。與上個實驗比較，我們發現 Merge 模組仍然與原始輸入的航線數量無明顯的相關。綜合 5.4 節和 5.5 節的實驗，我們知道 Merge 模組耗時的主要關鍵為資料集內的航線距離，也就是說如果航線經過的 cell 格數越多，則輸入 Merge 模組中的矩陣就會越大，耗時也會上升

表 5-6 不同航線數量下各模組的耗時

毫秒(ms)	BulidRoute	BulidCell	GetMajorVector	Merge	Display
20%(527 筆)	1.6203	0.0169	126.4589	6.157	113.6855
40%(1055 筆)	3.2277	0.0188	222.1784	7.4502	117.2568
60%(1582 筆)	4.3416	0.0184	323.6198	7.3958	146.6628
80%(2110 筆)	5.8684	0.019	461.5361	7.9113	195.5296
100%(2638 筆)	9.453	0.0175	534.4105	7.1474	296.2773

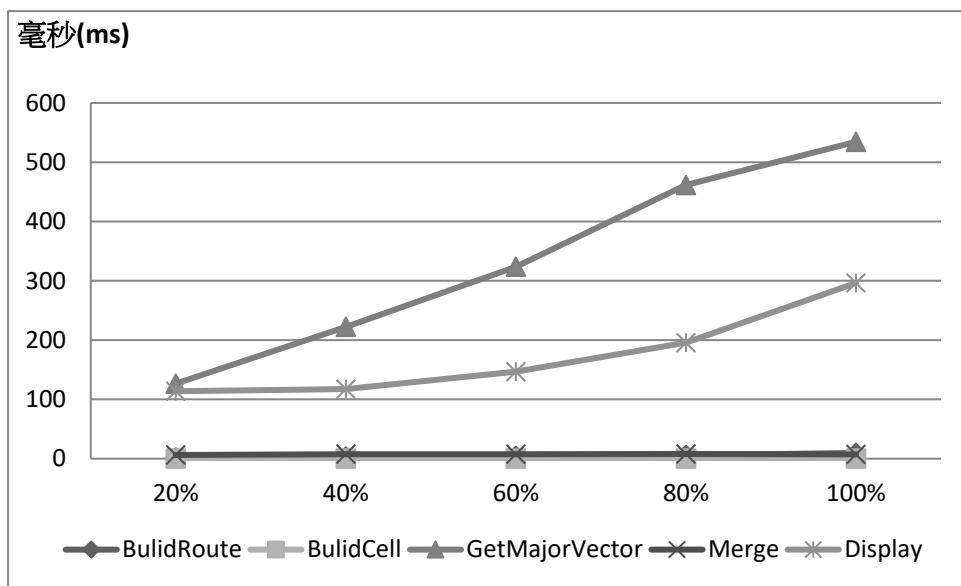


圖 5-10 不同航線數量的耗時比較圖

結合上述所有實驗結果，首先我們可以知道 BuildRoute 模組和 BuildCell 模組的耗時極低，幾乎對系統沒有影響。而 GetMajorVector 模組則使用單迴圈將航段與 cell 相交，得到每個 cell 中的平均線段，模組耗時最高，與 cell 格數大小和航線數量大小也都正相關。至於 Merge 模組，分別讀取 avgcell、link 與 turn 矩陣，間接與 cell 大小和資料中航線的長短相關，模組耗時對系統有小部分的影響。最後的 Display 模組使用單迴圈，顯示摘要化路徑，模組耗時僅次於 GetMajorVector。

## 第 6 章 結論與未來方向

在本論文中，我們從複雜的旅遊路徑資料中，找出主要路徑，此類摘要化路徑可以告知使用者，以往航空旅客最主要的移動方式有哪些。例如從第 5 章的實驗圖中，我們可以知道從台灣到歐洲的行程裡，在歐洲的義大利、德國與法國轉機的航線是最多的。

在研究中我們遇到了一些問題與困難，比如在 4.2 節我們提出了合併航段的想法，為了避免在 cell 中一次出現太多的路徑資訊，所以只主要合併 4 個大方向的航段(再輔加 4 個較少用到的方向)，總共 8 個方向。這個想法雖然成功解決了來自不同方向航段交錯後，線段過於雜亂的問題，但當一個地區作為密集中轉區，且中轉方向極多時，還是會發生路徑交錯雜亂的問題。例如圖 5-2 為台灣到歐洲的旅行路徑圖，地圖中的摘要化路徑前中段並沒有出現雜亂的資訊，但路徑末段由於在歐洲地區的中轉可能太多，所以路徑資訊呈現雜亂的情況。

其次是合併出來的平均線段有中斷的情況，在 4.3 節中，我們曾解釋平均線段中斷發生的原因，並提出了解決的方法，也就是記錄航線模式至 link 與 turn 矩陣後，並在 Merge 模組把中斷的線段連接起來，但這個方法會導致中轉點偏移實際位置。比較大的問題是摘要化路徑會中斷，我們在 5.2 節中曾解釋這些問題與發生的原因。主要是由於我們連接路徑的方法，是一格一格找出最多路徑的移動方式後將其連接，這會導致有些路徑在某處的合併數量突然高於或低於門檻值，而呈現路徑中斷的現象。

我們也進行實驗比較系統中不同模組的效率，發現路徑數量大小、資料中航線所移動的範圍與 cell 大小都會影響系統耗時。在模組中耗時最重的為 GetMajorVector 模組，耗時極為容易上升，而 Merge 模組的耗時則主要受到航線所覆蓋的 cell 格數影響。

最後關於本論文未來的研究方向如下：

1. 提高摘要化路徑的準確率，解決上述路線偏移和中斷的問題。
2. 加入其他資料進行更深入的分析，例如使用某個航線的旅客數。
3. 增加系統功能，使其更符合使用者需求，例如，可以彈性調整不同區域的 cell 大小，讓起點所在的 cell 範圍變大，迄點所在的 cell 範圍變小。

## 參考文獻

- [Pu06] Ken Pugh, “Prefactoring”, O'Reilly Media, 2006.
- [SMS11] Ricardo Silva, Jo~ao Moura-Pires, and Maribel Yasmina Santos, “Spatial Clustering to Uncluttering Map Visualization in SOLAP”, Proceedings of the International Conference on Computational Science and Its Applications conference (ICCSA), Part I, LNCS 6782, pp. 253–268, 2011.
- [GZJG12] Diansheng Guo, Xi Zhu, Hai Jin, Peng Gao and Clio Andris, “Discovering Spatial Patterns in Origin-Destination Mobility Data”, Transactions in GIS, 16(3): 411–429, 2012.
- [OSK14] Dev Oliver, Shashi Shekhar, James M. Kang, Renee Laubscher, Veronica Carlan and Abdussalam Bannur, “A K-Main Routes Approach to Spatial Network Activity Summarization”, IEEE Transactions on Knowledge and Data Engineering, 26(6): 1464–1478, 2014.
- [MSZ13] Elio Masciari, Gao Shiand and Carlo Zaniolo, “Sequential Pattern Mining from Trajectory Data”, Proceedings of the IDEAS conference, October, 2013.
- [WLLP14] Yu-Ting Wen, Chien-Hsiang Lai, Po-Ruey Lei and Wen-Chih Peng, “RouteMiner: Mining Ship Routes from a Massive Maritime Trajectories”, Proceedings of the IEEE International Conference on Mobile Data Management, 2014.
- [JQS14] Tao Jia, Kun Qin and Jie Shan, “An exploratory analysis on the evolution of the US airport network”, Physica A: Statistical Mechanics and its Applications, 413(1): 266-279, 2014.
- [WZ15] Wenjie Wu and Zhengbin Dong, “Exploring the Geography of China's Airport Networks: A Hybrid Complex-Network Approach”, Technical Report of the Spatial Economics Research Centre (SREC), SERCDP0173, 2015.
- [呂 09] 呂正中, “從小世界與無尺度角度探討全球城市網路之連結特性-以飛機航線及其機場據點城市為例”, 國立成功大學都市計劃研究所碩士論文, 2009.
- [李 13] 李函恩, “台灣西部主要機場之客運市場區隔”, 國立交通大學交通運輸研究所碩士論文, 2013.
- [林姚 15] 林鴻智, 姚立德, “基於 DBSCAN 演算法結合 Google Earth 對台灣落雷分佈研究”, 台灣地理資訊學術研討會論文集, 2015.
- [孫 10] 孫佩瑜, “應用敘述性選擇法分析低成本航空公司服務之願付價格：以台北-北京航線為例”, 國立嘉義大學行銷研究所碩士論文, 2010.

- [陳 13] 陳瑩霜，“航空公司綠色品牌形象與品牌關係品質之研究”，逢甲大學運輸科技與管理研究所碩士論文，2013.
- [謝 06] 謝淑芬，“空運學航空客運與票務”，五南圖書出版公司，2006.