趨勢科技
2013 騰雲駕霧 BIG DATA 創意程式大賽

Project Name

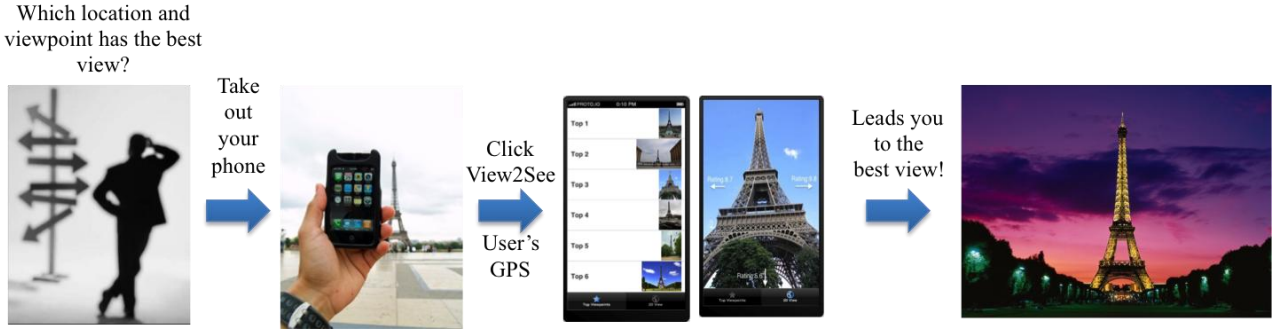
View2See

Team Name

Kuang Gong

1. 摘要

近年來，隨著國人旅遊風氣的盛行，坊間旅遊書、Blog 遊記、Google Map、旅遊景點推薦系統等廣泛受到大眾的喜愛，然而對於旅遊景點的最佳觀賞地點與視角，目前有賴遊客於旅遊前自行上網做功課，交叉比對網路上的資訊，並未有一套有系統的方式能輔助遊客快速搜尋喜歡的賞景地點，並即時導航至目的地。如下圖所示，本方案能提供遊客一個更便利的搜尋方式，減少自行探勘資料與在旅程中迷失方向所耗費的時間。



2. 問題或機會之定義

我們常遇到的一個困擾是，平時常在網路或報章雜誌上看到許多很美麗的建築物或風景照片，但當我們實際到當地旅遊，想要找到這些照片中的觀賞視角來欣賞，或是拍攝我們所看到相片中的景物時，卻苦無找到如此好的觀賞視角或觀賞時間。例如我們在網路上看到別人拍攝的一張巴黎艾菲爾鐵塔的照片(Fig. 1a)，當我們前往巴黎時也想到這個地點來欣賞鐵塔時，常遇到的情形可能有: (1)鐵塔距離我們太遠了，因此看不太清楚 (Fig. 1b) (2)鐵塔被其他遮蔽物給擋住了，使得我們無法欣賞景物的全貌(Fig. 1c) (3)我們可能太接近景物了，如果沒有廣角鏡，我們很難拍下整個鐵塔(Fig. 1d)。為了避免這種困擾，遊客必須在出發前做很多行前準備，藉由上網或是翻閱旅遊書籍來了解大家最推薦的觀賞視角，然而這些行前準備相當費時，對使用者而言可能是個負擔，另一方面，這些資訊也未必是最新的，如果有更好的新推薦觀賞視角時，我們也未必能從這些手邊的資訊得知。

旅遊相關軟體	View2See	旅遊景點推薦系統	Google Map
功能比較	View2See 會針對使用者想去觀賞的景點，推薦評價高的觀景地點或視角，並利用 3D 街景圖讓使用者更直覺地去瀏覽不同位置、視角的評分，再藉由 Google Map 的輔助，指引使用者前往絕佳視野的位置。	利用使用者現在所在的 GPS，去搜尋在這附近的熱門旅遊或是拍照景點。然而搜尋到的景點和使用者目前所看到的景點是不相同的，無法滿足使用者想要找到最好地點及視角來觀景的需求。	可以根據使用者現在的 GPS 來替使用者做導航，甚至提供街景圖給使用者做參考。然而它無法針對使用者目前想看的景點，來推薦最佳觀賞位置及視角。

因此我們希望能設計一個自動化的系統，讓系統會去搜集各景點的圖片，依照不同的拍攝位置及視角對這些照片做分類，再根據網路上網友們的評價，將最好的觀賞位置及視角推薦給使用者，系統同時提供 3D 模型，以更視覺化的方式讓使用者瀏覽各個不同觀賞的視角及網友們的評價。

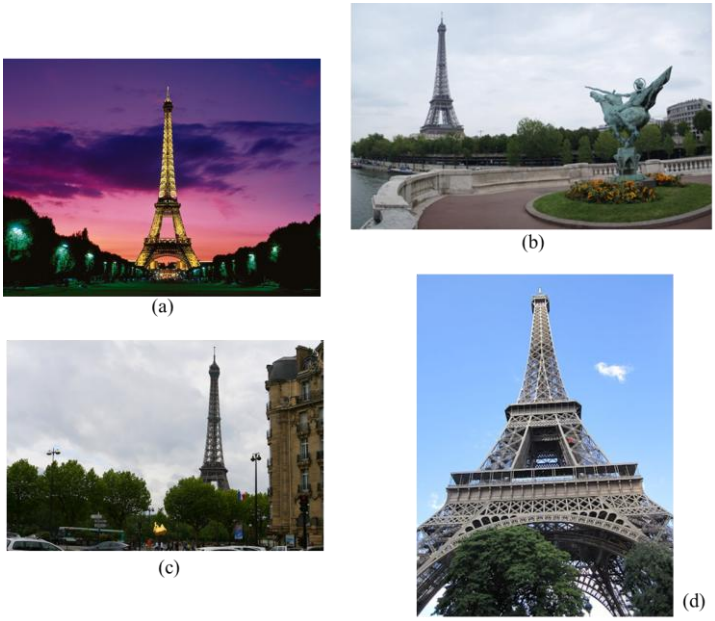


Fig. 1 不同位置拍攝的艾菲爾鐵塔

3. 術語

術語	定義
EXIF	Exchangeable Image File Format，專門為數位相機設計，可以用來記錄數位照片的屬性訊息和拍攝數據。
GPS	Global Positioning System，太空衛星全球導航系統，可提供所在地的經緯度座標以及海拔高度，最佳誤差約十公尺左右。
SIFT	Scale-invariant feature transform [1]，一種用來偵測與描述圖像局部性特徵的演算法，雜訊容忍度強，而且偵測效率高，處理速度接近即時運算，適合用在 Big Data 中進行快速且精確的配對。
GIST	一種用來偵測與描述圖像整體性特徵的演算法，執行效率快，而且可任意搭配使用低中高等級的圖像特徵。[2][10]
POI	Point of Interest，讓人覺得有興趣、有幫助的位置或地點。

4. 解決方案內容

回顧目前已有的解決方案：利用旅遊書、網路 Blog 的搜尋、搭配地圖來尋找有特色的地點，但對於如何快速找到確切的觀景位置、絕佳攝影視角並沒有有效的解決方案。而在影像處理上，目前已有成熟的技術，可利用相片推算拍攝的位置及視角，但缺乏實際的應用整合。

本方案藉由網路無遠弗屆的力量，大量探勘網路上對特定觀光景點(ex: 台北101、高雄愛河之心、台中公園湖心亭…)的影像，匯集成資料庫。根據相片的 EXIF 資訊，依照拍攝地點 GPS 位置與拍攝時間做分類，再利用雲端技術廣大的計算能力，分散式、平行化地進行照片特徵分析、視角估算、拍攝位置估計，還原建築物原貌、建立三維立體模型，並歸納整理照片中影像位置，選擇最具代表性的照片，提供使用者由不同位置觀賞景點之圖像參考。搭配 GPS 及 Google Map 導航的功能，可快速引導使用者如何抵達觀賞地點，本方案將能補充旅遊書、部落格的不足，並結合先進的影像處理技術讓使用者獲得更棒的使用經驗。

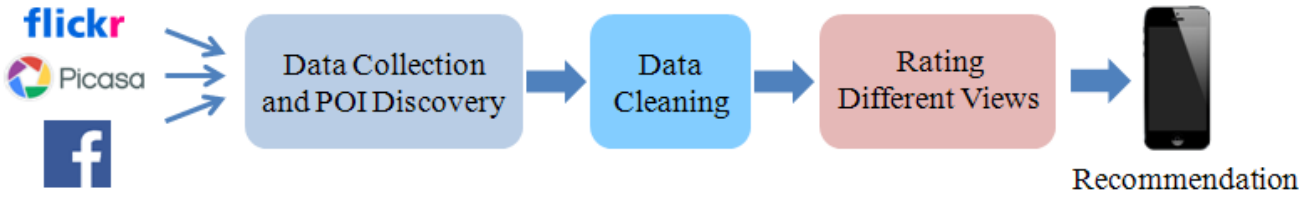


Fig. 2 系統架構圖

系統運算架構如上 Fig. 2 所示，首先透過 Flickr、Picasa、Facebook 等知名的照片分享網站，蒐集具有 GPS 資訊的照片，接著進行資料分類、過濾、評比等步驟最後總結出推薦的觀景地點，最後利用搜集來的照片們還原建築物、街景或景觀原本的 3D Model，利用圖形化的介面進行推薦，以下就各步驟的流程進行詳細的說明。

4.1. Data Collection and POI Discovery

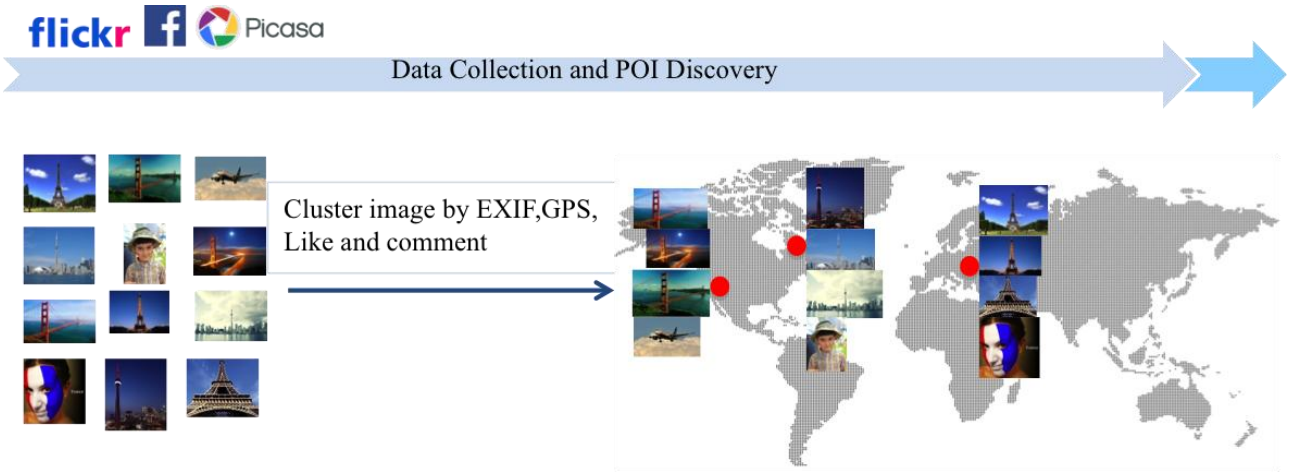


Fig. 3 Procedure of data collection and POI discovery

透過 Flickr、Picasa、Facebook 等知名的照片社群分享網站，收集大量且足夠的現存照片資料並記錄每張照片的 EXIF、Like 數量、comment 等文字資料，未來系統會定期地去收集新上傳照片的資料。收集到的這些照片資料，第一步驟先根據 EXIF 中的 GPS 資訊以及照片的景點名稱 tag，並使用[3]的作法即可將這些照片資料分類出所有景點的 cluster。

4.2. Data Cleaning

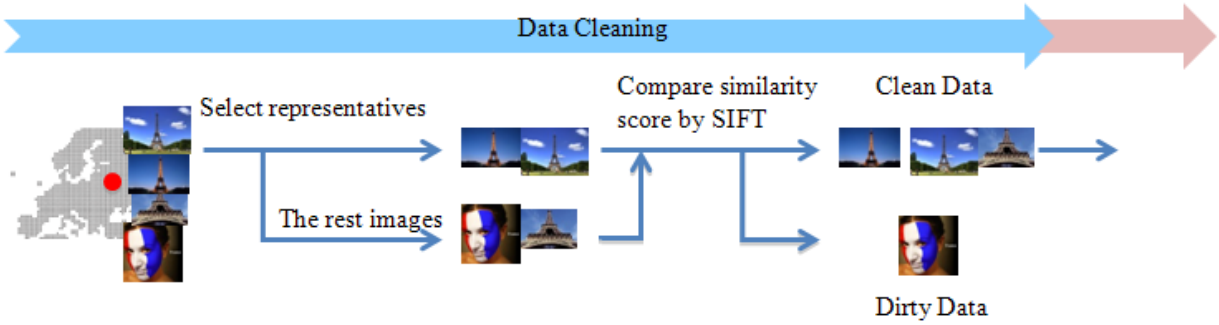


Fig. 4 Procedure of data cleaning

分類出所有景點之後，第二步驟我們必須針對每一個景點分別 mining 出它的最佳觀景攝影位置。我們可以根據某一景點的所有照片的 GPS 資訊再進一步做更細的 clustering，以找出在此景點中幾個比較熱門的觀景攝影位置，某個位置的 cluster 越大表示越多人在此地拍照，因此應該是較好較推薦的景色。

然而這裡將會遇到一個非常重要的問題，就是我們當初從網路上爬下來的大量關於此景點的

照片並不是 clean 的，舉例來說：當我們將所有"巴黎鐵塔"的照片從網路上爬下來時，並不會每一張照片都真的有照到"巴黎鐵塔"，或是因為人像太大而遮蔽了巴黎鐵塔。因此在這邊我們必須先進行 data cleaning，也就是我們想把不合格的照片給過濾掉。要使系統能自動做到此件事，一個方法是我們先找出一張非常 clean 的照片(清楚拍到"巴黎鐵塔")，我們稱之為 model data，並將剩下所有的照片都 extract 出影像內容的 feature(如利用 SIFT 此種可以 robust 抓出特徵點的 feature)來和此張 model data 的 feature 作 similarity score 的計算，當某張照片的 similarity score 小於某個 threshold 我們就將它視為 dirty data (並未清楚拍到"巴黎鐵塔")。

至於如何找出 model data，在這裡我們也提出一個自動挑選的方法，即是分析照片上的 comments 或是 like 統計數量資訊。舉例來說，有關巴黎鐵塔的照片中，若有某張照片出現很多其他網友們的 like 或是正面的 comments，則我們傾向確信這張照片一定是有拍到主角"巴黎鐵塔"並且拍得很漂亮，否則不會得到這麼多人的 like。而為了讓此方法更佳的 robust，在此我們不只選出一張 model data，而是會選出 N 張 model data (comments 和 like 統計數量排名前 N 名的照片)，剩下的不確定是 clean 或 dirty 的每一張照片都跟這 N 張 model data 做 similarity score 的計算，若此張照片和 N 張 model data 中的很多張照片的 similarity score 都非常小，則我們可以很有信心地將此張照片視為 dirty data 並過濾掉。

經過以上步驟後，任何一個景點的照片都可以自動被分為兩類：(1)確實拍到重點景物的照片，(2)沒有拍到重點景物的照片，將第二類的過濾掉後，剩下的第一類即較能確保它們是 clean 的 data。

4.3. Rating Different Views

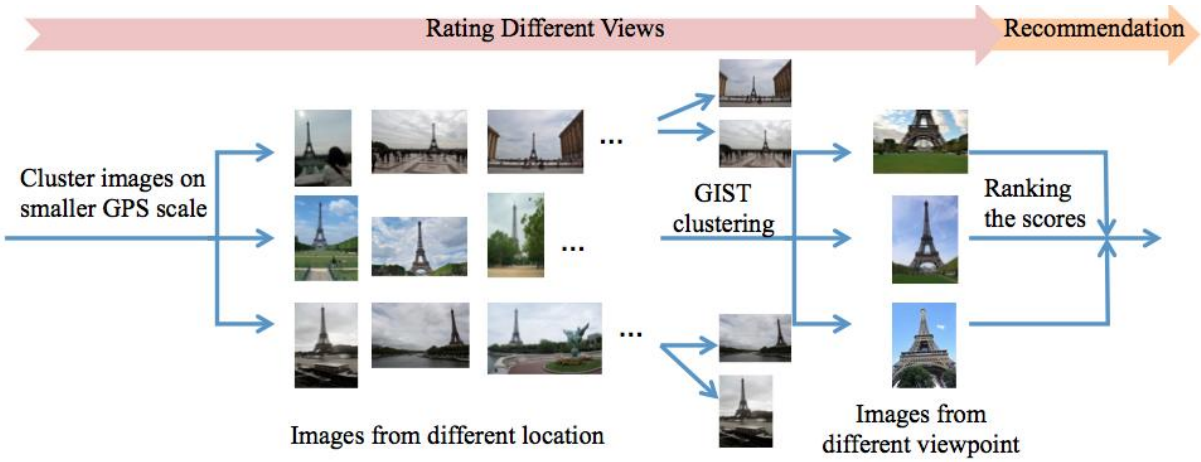


Fig. 5 Procedure of rating different views

上面的步驟已找出某重要景點周遭拍攝地點，在第三步驟中我們進一步細分 GPS 的位置，分類景點附近不同的賞景位置(Local GPS clustering)，如 Fig. 1。然而同一個位置也可能有不同的視角(ex: 地標在中間、靠左、靠右；仰角、俯角)，因此我們進一步根據不同賞景位置，利用圖像的 GIST 特徵[2][10]進行不同拍攝視角的分類(perspective clustering)，如 Fig. 6。分類完畢後，我們將對不同拍攝角度進行評比，評比會依以下標準先後進行排名：

1. 各 cluster 有被標為"Like"或是獲得正面評價的照片數目
2. 在該拍攝地點與視角拍攝的人數(照片張數)

(a) (b)

Fig. 6 同一地點不同視角的照片

我們假設越多人想拍照的地點也會是越佳的賞景地點，但為了避免最多人拍照的地方只是易達性最高的地點，而非最漂亮的觀景地點的問題，在評比的流程中我們將優先考慮照片被 Like 或獲得正面評價的數目，但為了避免親友捧場造成單張照片 Like 數目過份提高的狀況，我們僅考慮 cluster 中有獲得 Like 或正面評價的照片張數，而非 cluster 中照片所獲得的 Like 總數。以表 1 為例，在第一步的評比中的分數將是 B = C > A，在第二步的評比中則可進一步得知 B > C，因此最終獲得最佳推薦分數的會是視角 B，因為第一步獲得了最多的正面評價，表示較能引起他人認同；第二步因為照片張數最多，顯示此視角最能獲得他人的青睞。

表 1 範例資料

視角	照片張數	被 Like 的張數及正面評價數
A	40	1
B	20	5
C	10	5

4.4. Recommendation

第二步驟中，我們蒐集了同一景點周邊不同位置的照片；第三步驟中，我們為每個拍攝地點與視角計算了分數。在呈現推薦賞景地點前，從各視角的 cluster 中挑選出一張有代表性的照片，由 Photo Tourism 的技術，可利用幾張不同視角的 sample 照片，重建景物的 3D Model[4]。最終推薦時我們將提供兩種介面給使用者選用(1)列表(ex: Fig. 8a)，將所有視角依分數高低列表，讓使用者可以快速找到最佳的視角(2)圖形化介面(ex: Fig. 8b)，利用圖形化的介面具體呈現各拍攝視角的分數，使用者可透過點擊任意位置去瀏覽該視角的景觀，讓使用者更直覺得去瀏覽各視角與該視角的評價。最後由視角回推適合觀賞景點的位置，搭配 Google map 的 GPS 導航迅速了解該如何抵達該地點。另外我們提供時間軸，讓使用者可觀賞不同時段的景色(ex: Fig. 8c)。

Fig. 7 3D Model 範例

(a) (b) (c)

Fig. 8 照片推薦介面

我們這套系統必須架構在雲端系統上才有辦法運作，因為我們處理的資料包含了 Big Data 的四種特質：

Volume	(1) 系統必須獲取大量的圖片資訊，像是來自 Flickr, Picasa 等網站上的圖片，我們光搜尋台北 101 的圖片，Flickr 出現了超過 40 萬筆的照片，Picasa 也出現了超過 1,000 筆的照片，加起來約略為 4TB 的資料量。而世界上其他著名的觀光景點至少有上萬個，加上最近 Flickr 將每個使用者免費上傳空間提升為 1TB，因此一般的運算系統是無法處理如此大量的資料 (2) View2See 會截取影像的 feature 來做相似度的比對及 clustering ，然而 feature 的比對非常耗時，例如光是比較兩張影像的相似度，就大概花上 5 秒左右，因此如果用一般的單機電腦來比對上萬張的影像，一個景點就要花上超過一個月的時間來作比對，這效率是非常差的，使用者根本無法接受，所以就只能透過雲端平台的計算資源才有辦法達成
Velocity	目前的網路圖庫資料流量都很大，像是 Flickr，光是搜尋「單日台北 101」的照片，即可得到超過 1000 筆的相片，因此我們的系統必須持續分析新上傳到這些網站上的圖片，將最新熱門景點的照片截取至我們的資料庫做分析，以更新 View2See 提供給使用者的資訊
Variety	系統處理的資料包含了影像的資料、影像的 EXIF 資訊(如 GPS 位置、拍攝時間)、照片的 tag 資訊，同時必須去分析網頁內容來了解所有使用者對該張照片的評價數以及透過語義分析找出正面評價的詞語
Veracity	使用者上傳的照片可能包含錯誤的 tag 資訊，例如:有一張照片的 GPS 是巴黎的位置、tag 的內容也是巴黎，但其實這張照片的內容只是一張在巴黎拍的皮包;或是照片中有巴黎鐵塔，但照片中的人臉佔據相片中大部份的空間，這種照片也缺乏參考價值，因此我們必須透過 feature 的比對將 tag ,GPS 資訊和我們目標影像內容不同的照片給剔除掉

5. 解決方案之價值

Fig. 9 解決方案價值之分項

本系統的功能除了可以提供非攝影專業的一般使用者於旅遊時使用，還可以將其擴展至提供政府機構使用，甚至可延伸至公眾及公益性質，以下就這三項分別予以說明。

5.1. 一般使用者

5.1.1. 推薦絕佳觀景攝影地點

手機幾乎已經成為現代人隨身攜帶的電子產品，有了手機隨時隨地都可以拍下想要珍藏的畫面，而且因為硬體規格不斷地升級與進步，使得手機拍攝出來的照片，具有相當程度如相機拍攝的水準，誠如第二節「問題或機會之定義」段落的說明，這些照片可能因為拍攝的視角或時間的不同，而有美

感或質感上的差異，或是單純想要尋找最佳的觀景位置，以如 Fig. 10 五張台北 101 大樓的圖片作為例子，上排兩張是拍攝視角的不同，下排三張是拍攝時間的不同，不同的視角與時間確實影響了照片和當場觀賞的感覺，因此推薦拍絕佳的觀景或攝影位置，是本系統的第一個核心價值。

Fig. 10 台北 101 大樓攝影差異比較

5.1.2. 提供 3D 環視模型

本系統的第二核心價值在於藉由建立 3D Model，提供使用者更視覺化的瀏覽方式，以立體的圖形化介面讓使用者快速檢視特定位置視角的照片，而且因為是由多張照片分析還原的 3D Model，所以可以清楚了解周遭環境的景物及相對位置關係，讓使用者更能掌握自己所在的環境位置，對於指引方向上很有幫助。

5.2. 政府機構如觀光局

5.2.1. 宣傳特定景點及相關節日活動

本系統與旅遊景點有密切關係，因此可提供政府單位使用以宣傳旅遊景點及相關活動，例如觀光局想宣傳澎湖觀光，初始提供系統基本的澎湖相關景點的圖片和資料，之後系統將會從照片社群網站收集更多的相關照片，並建立起該觀光景點的完整系統資料；而如果想推銷節日活動如澎湖的花火節，則必須提供花火節的相關資料及含有該節日元素的照片，之後建立的系統資料照片以及 3D Model 即會與花火節有所相關，如此應用可透過本系統這個媒介，將觀光局等政府機機構的活動推廣給更多遊客知曉，提供更多的參考旅遊景點。

5.2.2. 取得遊客分布資訊

本系統的運作有賴於從網路上收集的大量照片資料，若反過來利用這個特性，根據各個景點的相關照片數量，可以推測出正相關的各個景點遊客數量的分部資訊，進而使得政府單位可針對各個景點提供或改善必要之硬體設備，提高景點的旅遊品質，例如當太魯閣相關的照片在短時間內有明顯的增加幅度，則可能透漏出太魯閣遊客數量的增長，進而需要增設公廁或觀景台等硬體設備以符合遊客的需求。

5.3. 公眾及公益性質

5.3.1. 記錄景觀環境變化

照片的 EXIF 是本系統收集資料的重要項目，其中的 GPS 和拍攝時間是會使用的重要依據資料，如果針對某個景觀收集夠多的照片，假如收集了一兩年之內的大量照片，那麼可根據月份或季份做出對應的系統資料及 3D Model，即可記錄該景觀的變化狀況，後續還可以提供政府單位評估是否要對該景觀環境採取適當的保護措施，例如自然景觀環境區域出現不明的違章建築、過度開發或天

災地震土石流等因素，使得 3D Model 產生明顯變化，即可提供政府進行違建取締或明確的環境變化警訊。

5.3.2. 提供環境變遷教育

如上段所述，收集某地大量的照片並以小段時間的間隔作記錄，可以觀察出環境的變化；如果將收集照片的時間範圍拉長至十年以上，不論是用收集現有的照片或系統長時間運作的記錄，可作為長時間該地變遷的證據，都市地區的記錄可以觀察到人口的發展以及都市更新計畫，自然景觀的記錄可以觀察到人為的入侵或氣候的變遷，這些長時間的記錄都是可以作為教育的好教材，Fig. 11a 所示為 Windows XP 的經典桌面圖，它拍攝的時間是西元 1996 年，而 Fig. 11b 右圖為同一地點於西元 2006 年時所拍攝，兩張照片很有時過境遷的對比感。

(a) (b)

Fig. 11 Windows XP 經典桌面圖十年變遷

6. 可行性分析

可行性分析我們會根據此解決方案的(1)技術可行性，以及(2)風險評估，兩部分來分別作討論。

6.1. 技術可行性

針對我們欲處理的問題可以就以下幾點來分析技術上的可行性：

6.1.1. Mining 出 POI Clusters

由於現今照片社群分享網站(Ex: Flickr, Picasa等)的蓬勃發展，我們可以取得大量user在各POI所拍攝的附有GPS經緯度資訊的照片，接著利用經緯度資訊作為feature對這些照片進行clustering，將可以得到許多大大小小的clusters，當一個cluster的照片數量多到一定程度的話，我們就可以把它視為是一個POI (因為當某個地方有很多人拍照的話，我們可以合理斷定它是個POI)。經過survey，這些照片社群分享網站皆有完善的API可供程式開發者使用，因此要取得這些data並不困難。另外由於資料量非常龐大，在這邊很有可能會遇到clustering會執行非常久甚至跑不完的問題，因此在這邊我們必須把clustering的演算法架在雲端平台上，而目前文獻上已經有一些clustering algorithm based on cloud computing可以直接拿來使用，如[5][6]，以確保可在合理的時間內做完clustering。POI這個部分主要是根據[3]的作法。

6.1.2. Mining 各 POI 中的最佳觀景或攝影的地點

由於 POI 通常會是一個大範圍的區域，因此針對每一個 POI 中的所有照片，我們可以進一步再利用經緯度資訊做更細的 clustering，如此一來將可以得到好幾個 clusters，而 clusters 中照片數量越多的就越可能是最佳攝影地點。這部分所需要的技術和 6.1.1.一樣，即我們需要利用雲端上的 clustering algorithm 來做有效率的 clustering。

6.1.3. Mining 各 POI 中的最佳觀景或攝影的視角

在一個 POI 中的同一個位置，也有可能會有各個不同的攝影視角，因此在這邊我們的做法

有兩個：

1. 統計出在此位置拍攝的照片中，哪些視角是被拍攝最多次的：
   要統計此資訊，我們直接分析照片的影像內容並取出每一張照片的 feature (如 GIST、SIFT)，接著利用這些 feature 將照片做 clustering。由於同一拍攝視角的相片，它們的 feature 會比較像，因此被分到同一個 cluster 中的照片基本上都是屬於同一攝影視角，所以我們只要統計各 cluster 中照片的數量，即可算出此視角被拍攝的次數。

2. 利用從相片社群網站爬下來的照片中的 comment 資訊：
   根據我們的觀察，通常一張照片的拍攝視角越好，其他 user 就會給予越多正面的 comments (如 good, beautiful, nice 等等)，因此我們可以藉由分析 comments 中的文字內容，來推算出此張照片的拍攝視角是否優良。

6.1.4. 建立 POI 的 3D viewing model

我們可以利用某 POI 的所有照片作為 Input，並利用[4]的作法建出 3D viewing model。然而這邊將會有一個困難點，也就是我們手邊擁有的 POI 照片 data 並不是完全 clean 的。舉例來說：當我們將所有"巴黎鐵塔"的照片從網路上爬下來時，並不會每一張照片都是很好的 input，有些照片可能有大量的人像入境，或是根本沒有拍到巴黎鐵塔。因此我們必須先做 data cleaning。我們的做法如下:

1. 首先我們必須先抓出幾筆 role model data。以例子來說，就是要先抓出幾張我們有信心是非常標準的巴黎鐵塔照片(沒有人像入鏡、有拍到巴黎鐵塔…等)，詳細自動抓取 role model data 的方法呈現在 4.2 的部分，在此我們會抓出不只一張的 role model data (N 張)。
2. 對每一張照片都根據影像內容來取出 feature，並跟所有的 role model data 都算出 similarity score，當 similarity score 大於某個 threshold，我們即可把它視為 clean data 並拿來使用。而由於照片數量非常龐大，必須做非常多次的 feature extraction 和 similarity computation 運算，因此這邊我們可以利用 map reduce 的架構(如 Fig. 12 所示)來實現有效率的 feature extraction 以及 similarity score 的計算。以下分別敘述我們 mapper 和 reducer 分別作的任務：

   Mapper: 每一個 Mapper 都會被分配負責做一張照片和 N 張 role model data 的 similarity score 的計算，並把結果 output 成 key | value 的形式。Key 為被處理那張照片的 ID，而 value 則為 N 個 similarity score。真實情況下由於 Mapper 數量是有限的，因此假設有 P 個 Mapper 且共有 M 張照片待處理，則每個 Mapper 需要負責處理 $\frac{M}{P}$ 張照片。

   Reducer: 上一步驟中得到的每個 key | value pair 首先會根據 key 的值被分到對應的 Reducer，而每個 Reducer 再根據 value (N 個 similarity score)來判斷此張照片跟那 N 張 role model data 中有幾張是相像的，若相像的張數大於某個 threshold (例如 $\frac{N}{2}$ 張)，則 reducer 便會把此張照片視為 clean data 並 output 出 ID | 1，其中 1 表示此張照片是 clean 的；反之，則 output 出 ID | 0。

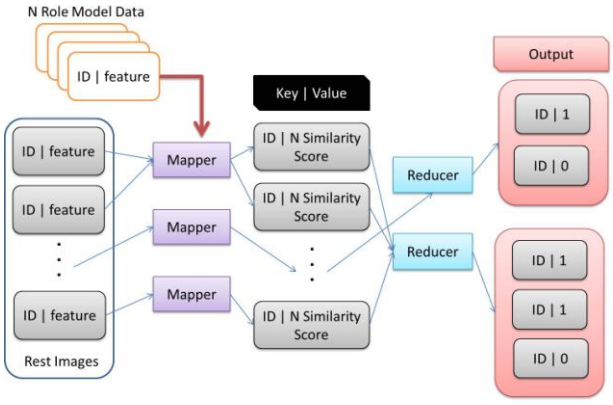


Fig. 12 Data Cleaning 的 Map Reduce 架構

相關 MapReduce 參考文獻：[7][8][9]。

6.2. 風險評估

除了技術方面的可行性之外，我們的解決方案還要在真實世界中是可行的，也就是必須考慮當系統在實際運作時會遇到的一些導致系統無法正常運作的狀況該怎麼處理，以下三種情形是考慮到我們系統在運行時，可能會遇到的風險以及可能的解決方式：

6.2.1. 使用者 query 的地點不存在於現有的資料庫

會發生此情況主要是因為該地點缺乏 geo-tagged 照片，因此當初並未被我們的系統爬到。若我們直接 return 空的結果給使用者，很可能導致使用者對於服務的不信任，進而導致我們的服務損失客群。因此在此情況我們會即時利用 user query(user 的 GPS location)丟到 Google maps API 中找到對應的地點名稱，並利用此地點名稱再回去 Flickr, Picasa 等照片分享網站爬照片(不管是否有 geo-tagged)，並直接利用將我們原先系統流程中"根據 GPS location 做 cluster 以找出推薦的攝影位置"的步驟省略，直接計算出"推薦的拍攝視角"以呈現給 user。

6.2.2. 使用者 query 的數量太多

當太多使用者在短時間內密集的 query 時，系統很有可能沒辦法負荷導致服務停止。在此我們可能解決的方式是將一段時間內的 query 以及相對應的計算結果做 cache，如此當下次相近的 query 進來時系統便不用重新再計算，直接 return 在 cache 中的結果即可。

6.2.3. 資料庫的 scalability 問題

由於每個地點每過一段時間都會有新的拍攝照片被上傳到 Flickr, Picasa 等照片分享網站，而我們的系統也必須持續地將這些新照片爬下來以使資料庫的資訊保持是在最新的狀態，然而這些資料庫中照片的量將會以非常快的速度成長，使資料庫很快就負荷不住進而導致需要不斷地重新規劃資料庫或是添增儲存裝置，如此一來系統的成本將會大幅增加。為了避免此一情況，隨著照片數量越來越多的情況下，我們在 data cleaning 步驟可以把條件訂得更加嚴格(例如將 threshold 提高)，使得合格的照片(clean data)維持在一定的數量，此外這也是個一石二鳥的做法，因為在 data cleaning 條件更嚴格的同時，所篩選出來的 clean data 會越 reliable，而最終 return 給使用者的結果的 quality 也會越好。

7. 參考

[1] Lowe, D. G. Object recognition from local scale invariant features. In Proc 7th Int Conf on Computer Vision, Corfu 1999.

[2] A. Torralba. Contextual priming for object detection, IJCV 2003.

[3] X. Lu, C. Wang, J. M. Yang, Y. Pang and L. Zhang, Photo2Trip: Generating Travel Routes from Geo-Tagged Photos for Trip Planning, ACM 2010.

[4] N. Snavely, S. M. Seitz and R. Szeliski, Photo Tourism: Exploring Photo Collections in 3D, ACM SIGGRAPH 2006.

[5] Jeffrey D. and Sanjay G., MapReduce: Simplified Data Processing on Large Clusters, USENIX Association OSDI 2004.

[6] Lizhe, W. et al., G-Hadoop: MapReduce across distributed data centers for data-intensive computing, future generations computing system 2013.

[7] Hadoop Image Processing Interface (HIPI) http://hipi.cs.virginia.edu

[8] Mohamed H. Almeer, Cloud Hadoop Map Reduce For Remote Sensing Image Analysis, Journal of Emerging Trends in Computing and Information Sciences 2012.

[9] Wang S.Y. et al., Learning by expansion: Exploiting social media for image classification with few training examples, Neurocomputing 2011.

[10] Auda Oliva, Gist of the Scene, Computational Visual Cognition Lab, MIT 2005.