Scalable Crowd-Sourcing of Video from Mobile Devices

一、 動機

在過去十年中，HUDs(頭戴顯示器, Head-up displays)已在軍事用途中普遍地被使用。然而隨著時代的演進，Google Glass 破壞式創新的出現，使 HUDs 以提供一般大眾日常使用爲目標前進了一大步。

HUDs 通常會整合許多影像相關感應器，令使用者能藉由 HUDs 隨時隨地拍攝第一人稱視角影片，並能即時將這些資訊分享至網路上。作者認爲若有許多人皆能使用如此設備，影片數量將大幅提升，或許將能帶來大量利益與優點(包含商業分析、公共安全等不同領域)。想當然耳，若欲使大家皆願意分享影片，需要有足夠的誘因與隱私，同時也需要考慮到大量上傳對網路頻寬所帶來的負擔。

在另一方面，穿戴式裝置的功能越趨強大、爲大眾提供更多便利應用的同時，該如何整合與影片相關的 sensor 資料，包含 gaze tracking、音頻、地理位置、加速度、生物特徵(如:心跳率)等等，以及該如何建置有價值的大規模可搜索的影片庫，並且讓使用者進行基於內容的深入搜索等，皆是未來所要關注的議題。

爲了因應此項科技的發展趨勢，本篇作者提出了分散式網路架構以取代傳統的中心化架構，試圖解決巨量資料所帶來的運算、儲存、隱私、頻寬等問題，同時亦實作出 prototype 以證明作者想法的可行性。

二、 議題

歸納作者所提出的大規模可搜索影片庫可針對以下兩面向的議題做爲探討：

(1) 使用者動機：

雖然現代網路使用者會自行上傳影片至 YouTube，然而這些影片通常與個人成就（如:第一人稱視角運動），或他們想分享的特殊場合（如:婚禮現場）相關。因此，縱使影片內容擁有足夠的個人利益做爲讓使用者花費時間去編輯和進行 tagging 的動機，但這樣的自利行爲仍然並不足夠使個別用戶願意持續地捕捉每天的生活場景，甚至共享對他人可能有其他價值的場景。因此，在缺乏個人動機的情況下，需要有一個更明確的激勵模型**(Incentive Model)**。

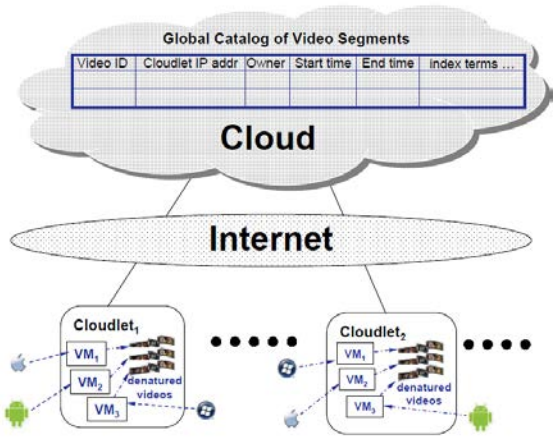
而基於上述缺乏動機的情況，作者於本文中提出了強健且可擴展的方法：透過讓使用者和服務提供者建立商業關係，如同圖書發行商與圖書作者，從而建立了一個簡單的激勵模型，令使用者得以透過拍攝或共享其他人認爲有價值的影片，以獲得財務獎勵。此外，作者也希望添加影片內容標籤的方法，能有一個基於內容的視覺演算法自動處理，避免用戶對日常生活影片手動進行標籤感到過於麻煩。

(2) 隱私問題：

穿戴式裝置的推廣令使用者得以隨時隨地錄製影像或其他資訊，並發布至公開網路上，而由於拍攝 always-on video(持續錄製影像)可能會有很多不可控制的事情發生，這些被分享到網路的內容很可能包含他人隱私內容，如：在拍攝場景時，一些使用者不想公布的物件或人可能在場景中出現。因此，在設計影像處理系統時，也需要考慮該如何對原影片內容進行後製或其他處理，令使用者能夠編輯或模糊場景中的個別物件(如：透過臉部辨識影像，將人臉打上馬賽克)。

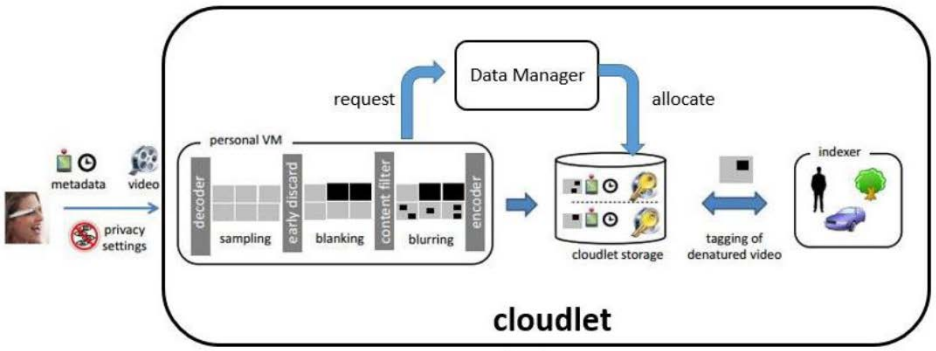
而因爲每部影片皆需要由使用者決定其中該被刪去的物件，但在編輯連續拍攝的影片需要花費相當多的時間。因此，當影片被上傳用於共享時，需要一個自動化連續執行這種編輯過程。在本文中作者提出了「**Denaturing**」，透過謹慎地分析拍攝影片與片段截圖，利用電腦視覺演算法進行臉部或物件的偵測與識別，或是對一連串的影片進行活動識別，再依據使用者隱私設定來進行模糊化或刪除。期望該系統能同時滿足不同使用者的不同隱私設定，倘若該機制太寬鬆會引起注重隱私的使用者不滿，但若太嚴格又會使影像價值降低，因此需在隱私與價值間找到平衡。

三、 研發方向



面對大量用戶共同分享的行爲，需要設計相應機制，避免網路過載並解決高上傳帶來的高資料傳輸問題。本文作者借用了過去 CDN 的設計概念，提出一套「**Gigasight**」雲端架構，如左圖所示爲一種反向 **CDN** 的混合雲架構，爲由多個「cloudlets」個體所組成的分散式雲端架構。Cloudlets 起源於雲端運算與行動運算的匯流，在每個 cloudlets 上會利用客製化的虛擬機(personal VM)來模擬運算程序的執行環境，使每個分散在各地的 cloudlets 皆能成爲一個運算節點。此架構模仿行動運算的 CDN 架構：device-cloudlet-cloud，其中有一個 cloudlet 做爲中間的 data center。如同系統架構圖，cloudlet 可以做爲 data center 儲存使用者的 denatured videos，僅將影片的 metadata 上傳至雲端，以此降低雲端的流量負擔。同時，cloudlet 也負責進行 denature 等圖像處理任務，將此類任務放置 cloudlet 處理將比於行動裝置處理的效能更佳。

當使用者透過行動裝置收集到原始影像後，這些資訊會被上傳至距離最近的 cloudlets 進行 denaturing 運算(亦即 offload)。如下圖所示，於 cloudlets 運算完畢後會向 Data Manager 發出請求，由其替完成 denaturing 的資料，分配儲存空間，最後利用 denatured file tag 建立 cloudlets 上的 content indexer。在大多情況下，cloudlets 僅會將完成 denatured file 的 metadata(如:時間、地點)上傳至 cloud 做爲未來搜尋該內容時的辨識資料，而原始檔案將會被儲存在 local cloudlet 上。利用 cloudlet 除



了得以代替傳統中心化設計的 cloud computing，解決大量資料流問題外，同時也能提供較多的資安、隱私保護機制。

文中所提出的 GigaSight hybrid cloud 架構，用於從行動裝置進行可擴展的 crowd-sources 影片服務，架構如同上圖可分爲五個部分：

(1) Mobile Client：

考量到 HUD 爲了保持舒適度，重量尺寸有嚴格限制，因此儲存能力與處理能力也將受限。因此作者改由智慧型手機做爲 HUD 代理，當作是影片傳送到 GigaSight 架構的其餘部分緩衝。作者利用 App 設定錄影的隱私權限(包含何時何地的影片可以分享、人臉模糊等)，當手機連接至 Wi-Fi 時，立即將拍攝影片與其相關訊息(如:GPS、timestamp、sensor)上傳至 cloudlet，而隱私權限也會被用來在後續行爲進行 denaturing。

(2) Personal VM：

Personal VM 是存在於 cloudlet 上具有更多的計算和儲存資源的 GigaSight 組成要素，負責根據使用者於 mobile phone 上定義的隱私規則，對上傳的視頻進行 denaturing。當 mobile phone 發出其具有準備上傳的影片時，Personal VM 向 Data Manager 請求進行影片儲存空間分配，而後對其進行 denaturing。每個使用者分享影片時都會有個人的 VM，VM 會存於 Cloudlet 中，並會隨著使用者移動地點，自動切換至離使用者最近的 Cloudlet。

Denaturing 爲多步驟的 pipeline。首先，因爲若要使用原本的影片進行 denaturing 則會太耗費計算資源，因此在第一步驟中會選擇要用以進行實際 **denaturing** 的影片影格 **subset**。而後在第二步驟中，應用具有低計算複雜度、基於 metadata 的過濾器進行篩選，如：根據時間和位置值，決定該影片影格該被完全刪除或不需經任何修改。在第三步驟中，應用更複雜、基於內容的過濾器進行篩選。目前作者的 GigaSight prototype 僅支援臉部偵測，任何在影格中偵測到的臉部將會被進行模糊化。最後輸出一個 low-framerate(縮圖)的影片檔案，儲存在 cloudlet 的儲存系統上，做爲之後 indexing 和搜尋影片內容時的一個代表性的檔案。

而原本的影片經過 AES 加密後也會儲存在 cloudlet 上，當用戶想要看到完整的影片時，便會在原始拍攝影片的用戶個人 VM 內部對原始內容進行解密和 denaturing，若先前已經被搜索過的結果則將不重複此過程。

(3) Data Manager：

Data Manager 在 cloudlet 上的單獨 VM 中執行，負責管理 cloudlet 中影片的儲存和 metadata 資料庫。實作上，透過由 TastyPie、Django 和 Python 實作的 REST-API 來實現，並將不同影片的 metadata（如：影片的 ID、拍攝時間、影片長度、訪問控制權、GPS 座標等）儲存於 MySQL 資料庫中。

而當由 Personal VM 輸出的 low-framerate 的影片檔案被 Video Content Indexer 進行 indexing 後，輸出的 tag 結果將會含有影片中所含的物件以及這些物件在影片中的位置，Data Manager 會將這些結果儲存在資料庫中的個別 table 中。

(4) Video Content Indexer：

經過 denaturing 影片的 indexing，是由 cloudlet 上單獨 VM 執行的背景活動。每個 denaturing 的影格會由 computer vision code 單獨分析後產生該影格的 tag（在未來希望能分析影格序列並標籤出如鼓掌、擁抱等有意義的動作），並在 tag table 中建立 entry 儲存之。舉例來說，當條目內容爲「dog，zoo.mp4,328,95」，代表 indexer 在 zoo.mp4 中第 328 影格中檢測到狗，且具有 95%信心水準。此外，這些 tag 也會傳播到 cloud 上的 global catalog

中。

除了上述的 tag 外，爲了避免由於 denaturing 在 indexing 前操作使得一些 tag 沒有被標示出，因此，在 denaturing 過程中也會產出一些 tag，最後 tag table 上的 entry 將會是 denaturing 過程產生的 tag 以及經過 computer vision code 產生的 tag 集合。

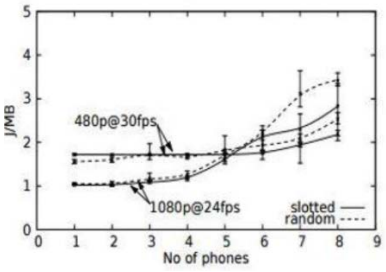
(5) Search Workflow：

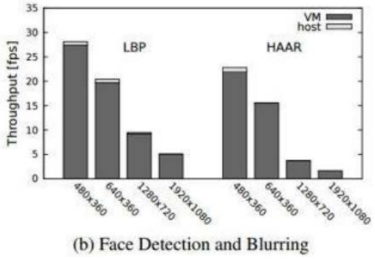
GigaSight 使用兩步驟的搜尋流程來幫助使用者找到內容相關的影片。首先，使用者會在 cloud 的 global 目錄上執行 SQL query，並得到包含 video segment 和 low-framerate 影片檔案的搜尋結果列表，並且得到這些 video segment 所在的 cloudlet IP 位址。第二步驟會進行基於實際內容的過濾，以得到更加相關的結果(Early Discard)。此步驟所需進行的計算量非常大，因此可進行平行運算、加快處理速度。

舉例來說，假設使用者要搜尋「昨天下午 2 點和下午 4 點之間兩個在一充滿了黃色的球的房間的小孩，且其中一個小孩穿著藍色格子襯衫」，在第一個步驟中將透過時間和「臉」這個 tag 來縮小搜尋範圍，並得到一個搜尋結果列表，該列表在第二步驟中會利用一些濾波器來過濾結果中包含黃色和藍色的結果，並將過濾後的結果回傳給使用者。

此外，作者利用 OpenCV 與 Android SDK 寫了兩支人臉辨識程式於智慧型手機上，對同一份 1080P 影像進行辨識，並記錄效能表現。如右圖所示可發現 GigaSight 採用的 offload 有較好的效能表現，而利用 cloudlets 進行 denaturing 也有較好的表現。不僅如此，根據網路科技的演進趨勢並配合網路理論推導，可得出 smaller edge cloudlets 相較於 larger edge cloudlets 爲較好的部署方式。

	throughput	Energy consumption
OpenCV	0.05 fps	50.46 J
Android SDK	0.31 fps	9 J





(b) Face Detection and Blurring

四、 未來預測

- **Computer Vision Algorithms 的改善：**

  由於 GigaSight 架構的表現非常仰賴於 denaturing 和 indexing 所使用的 computer vision algorithms，而該演算法的執行需要非常多資源，其效能的好壞會影響整個系統運作時的好壞。然而，隨著 GPU 虛擬化技術逐漸成熟，GigaSight 的 throughput 將可顯著的提高。

- **Cloudlet 上的儲存系統管理和資源回收：**

  在 GigaSight 架構尚未考慮到 cloudlet 的儲存空間大小問題，這個空間大小必須取決於影片的平均保留期、使用者的歷史使用經驗及使用者移動的軌跡（當使用者移動了特定距離，他所屬的 personal VM 會 migrate 到離該使用者較近的 cloudlet）來決定。爲了縮短回應時間，Cloudlets 通常也扮演著 proxy 的角色，欲儲存多久便是另一個議題，而這可以參考許多的 CDN 研究所產生的演算法。

- 影像資料品質對 **denaturing** 的影響：

  影像資料的解析度會對 denaturing 的結果造成影響，尤以臉部辨識影響最大。在較低的解析度下，因爲畫面模糊也使影像辨識準確率下降，然而系統難以限制使用者上傳的影像品質，或許增強演算法是另一個值得施力之處。