雲端即時監視系統火災逃生規劃研究
Simulate evacuation planning of fire disaster on the real-time cloud surveillance system

郭忠義、李秉祁、賴岱佑
國立臺北科技大學資訊工程系
Department of Computer Science and Information Engineering,
National Taipei University of Technology
Email: jykuo@ntut.edu.tw, sucdekey2@gmail.com, deyu1978@gmail.com

摘要

傳統的火災偵測多為溫度感應，但是溫度異常也可能會誤報，且傳統的火災偵測多為發布警報，然後以建築物內的號誌燈當作逃生建議路線，無法根據當時的火災情況做出建議的逃生路線。因此本研究實做一個雲端監控火災與逃生路線規劃系統，使用 IP Camera 搭配雲端系統監控火災，若偵測到火焰則立刻散佈火焰資訊；使用者的智慧型裝置安裝本研究製作的 app 程式，藉由網路先定位出該裝置位於何處，且該 app 程式會一直索取火焰資訊檔案，一旦接收到火焰資訊，app 程式將會依據定位的資訊，立刻模擬出當時的情境，且根據火焰的位置，隨時提供建議的逃生路線給使用者，讓使用者能盡速逃離現場。

關鍵字：火焰偵測、雲端平台、建築模擬、逃生路徑、Unity3D、A-Star 路徑演算法

一、緒論

隨著建築現代化，無論身在何處，都能見到許多大樓，不論是公司、學校、還是住宅，所在之處都會有大樓存在。火災總是出奇不意，就在人們正在上班、上課、研究、甚至是休息時，一場火災可能就發生了。此時，要在錯綜複雜的大樓進行逃生，除了需要熟悉地形，也需要判斷前方是否能安全通過。

根據內政部消防署全球資訊網的統計，104 年發生的火災次數以及地點如表 1 各類火災統計表，可發現火災地點為建築物的比例最大。

表 1 各類火災統計表

	火災次數	百分比(%)
建築物	1242	72.9
車輛	234	13.7
森林田野	60	3.5
船舶	10	0.6
其他	158	9.3
合計	1704	100

過去的火災偵測研究有粒子檢測、空氣透明度檢測與溫度感測[1]，感應到空氣、溫度的異常，並啟動警報鈴聲，逃生路線卻只有號誌可以引導人們疏散，但是如果建議的逃生路線上因為火焰而無法通行，就有可能無法逃生，且在火災中，人員傷亡最大的因素就是逃生路徑的設計不良所造成的[2]。為了保障人員的生命安全，即時監控火焰，並於第一時間提供使用者逃生環境及路徑，有助於將災害損失降至最低。

為了解決此問題，本研究實作一個雲端監控火災與逃生路線規劃系統，建立雲端平台、監控火焰，並使用 Sketch Up3D 及 Unity 3D 模擬建築物及規劃逃生路線，以利火災發生時將人員的傷亡降至最低，保障人員的安全。

在伺服器上建立雲端系統，架設 IP Camera 並監控火焰，將偵測到的火焰資訊存取在雲端平台，並由伺服器端通知所有用戶該地點發生火災，並規劃逃生路線進行逃生。逃生路線則使用 3D 建築物模型，模擬該建築物內部的構造，並利用程式計算出建議的逃生路線，幫助用戶進行逃生，其流程如圖 1 研究流程圖。

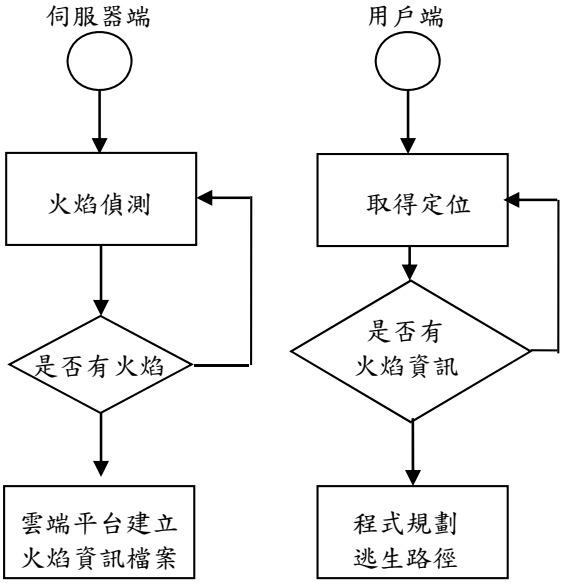

圖 1 研究流程圖

本研究以國立臺北科技大學的科研大樓為研究範圍，模擬 IP Camera 監控偵測到火災，及發生

時火災的逃生路線規劃，期望能應用於真實的情況。

本研究的組織如下，第二章內容為本研究相關文獻探討，第三章描述研究方法，第四章則為本研究的系統實作，第五章為實驗結果，第六章則為最後的結論。

二、相關文獻探討

2.1 火焰偵測

Celik 等人[3]的研究建立一個基於顏色而檢測的方法，使用 YCbCr 色彩空間比 RGB 能夠有效地將亮度與色度分離，使用模糊邏輯有效地分類出火焰與非火焰物件，而此方法的準確率高達 99%，而此方法的缺點也因為只用到顏色偵測火焰，因此誤偵測率為 9.5%。

Wang 等人[4]提出基於顏色並查表的方法(dominant flame color lookup table, DFCLT)，並使用三種色彩空間模型，RGB、HSV 與 CIE LAB。實驗結果有 90%的準確率，但是缺點為對於假的火焰規則或測量方法無法過濾之。

2.2 模擬建築工具

Maya 是一套 3D 建模工具，由 Adobe 公司開發，擁有 3D 動畫設計多數的功能。Maya 擅長光線與材質的設計，適合做美工、電影特效等，美工設計上，能添加多個材質並不會產生奇怪的顏色，可做得較為精緻[5]。

3D MAX 是一套 3D 建模工具，由 Autodesk 公司發行，雖然美工不及 Maya，但是構造圖之間的轉換卻較有優勢，比較適合部門分工時使用[6]。

2.3 逃生路線規劃

Dijkstra 演算法是由荷蘭電腦科學家 Edsger Wybe Dijkstra 提出。Dijkstra 演算法使用了廣度優先搜尋解決非負權有向圖的單源最短路徑問題，演算法最終得到一個最短路徑樹。

雖然 Dijkstra 演算法可以保證找到最短路徑，但在演算效能上有待加強。為了改良其演算速度，而有了 A*樂徑演算法。

A*路徑演算法運算的節點數量比 Dijkstra 演算法較少，可較快找到一條路徑，因此 A*路徑演算法的演算速度較快。A*路徑演算法能達到如此令人滿意的結果，是因為它採用了一套特殊評價公式(Heuristic Estimate)，此公式避免了一些不必要的路徑排除，所以能用較高的演算效率計算出一條最佳結果[7]。

A*路徑演算法以目前節點，利用公式計算附近鄰居節點的評價分數，並將鄰居節點加入串列，再從串列內挑選出評價分數最小的節點當作目前節點，直到計算出終點為止。完整的 A*路徑演算法如公式 1。

$$F(n) = G(n) + H(n) \qquad (1)$$

其中 n 為目前節點，F(n)為目前節點的評價分數總和，G(n)為從起始點到目前節點實際移動的距離，H(n)為目前節點到終點的估算值。

評價公式則是採用曼哈頓距離(Manhattan distance)，此距離有著極為簡單的公式，如公式 2。

$$H(n) = |x_1 - x_2| + |y_1 - y_2| \qquad (2)$$

x_1 是第一個座標的 x 值，x_2 是第二個座標的 x 值，y_1 是第一個座標的 y 值，y_2 是第二個座標的 y 值，即使第一個座標與第二個座標之間不是水行或垂直的，也是以水平或垂直線的距離去計算。

三、雲端監控火災與逃生路線規劃系統設計方法

本研究設計規劃有效的火焰偵測演算法，整合雲端環境。其次建立建築物模型，並實作出逃生路徑。最後在實作出定位程式，完成雲端監控火災與逃生路線規劃系統。

3.1 火焰偵測

為了使用 IP Camera 進行火焰偵測，本研究使用影像監控的方式偵測是否有火焰。首先使用 Sobel Filter 及 Flame Texture 減少影像的運算量，再建立 Color Model 過濾非火焰區域，建立候選火焰區域並產生穩定的火焰區域，最終完成火焰物件。

3.2 雲端平台

傳統監視系統能較少能配合智慧型行動裝置，且傳統監視系統的硬碟常因長時間循環錄影而損壞，災難總是出奇不意的到來。而雲端平台無論在運算資源、儲存空間上都具有彈性。Chu 與 Wu[8]的研究使用雲端計算逃生路線，一旦發生火災，使用者藉由智慧型行動裝置與 RFID 閱讀器，發送至雲端伺服器，並回傳逃生路線。此系統能處理大規模的範圍，且雲端系統的成本也較低，也易於管理。

3.3 建築物模型

Sketch Up 3D 是一套相關專業人員常使用的 3D 建模程式。Qiao 等人[9]的研究顯示 Sketch Up 3D 有六大特色：室內設計、景觀設計、建築設計、即時渲染、可用性與檔案轉換便利。基於方便使用的理念，它擁有一個非常簡單的介面。Sketch Up 3D 世界中一個眾所周知的特性便是 3D Warehouse[10]。使用者可以利用他們的 Google 帳戶來上傳建立的模型，並且瀏覽其他的元件和模型。

Unity 3D 是一個用於建立諸如三維電動遊戲、建築視覺化、實時三維動畫等類型互動內容的綜合型創作工具[11]。Unity 3D 類似於 Director、Virtools 或 Torque Game Builder 等利用互動的圖型化開發環境為首要方式的軟體其編輯器執行在 Windows 和 Mac OS X 下，可發布遊戲至 Windows、Mac、Android 或 iOS 平台。Unity 3D 的主要特色如下：

(一)內建地形編輯器，可快速製作地板效果

(二)內建 PhysX 物理引擎，提供重力、碰撞等效果

(三)內建粒子系統，可製作火焰及其特效
(四)支援多平台開發功能，可發布至 PC、Mac、Android、iOS 等跨平台

Ribeiro 等人[12]的研究使用 Unity 3D 建立一個火災疏散模擬環境。使用 30 個人當作樣品進行疏散測試，在最短時間內離開大樓。研究結果顯示可有效的進行疏散模擬，適合本研究的火災逃生模擬，將 Sketch Up 3D 所製作的模型匯入 Unity 3D 並進行火災逃生模擬。

3.4 逃生路徑

Hu 等人[13]研究如何在 Unity 3D 內使用 A*路徑演算法，先以腳本的方式建立「astarpath.cs」，此為 A*路徑演算法的核心，也是最重要的部分。而第二重要的是「Seeker.cs」，將需要尋路的物件(起點)放入此腳本，會自動將避開範圍內的障礙物並自動找到終點。最後還有些腳本，將修飾 A*路徑，使其能夠簡化運算。

本研究為了加速 A*路徑演算法的演算效率，使用了改良版 A*[14]。在原本的 A*路徑演算法添加了使用向量內積的作法，應用向量內積進行方向的判定，大於 0 代表同方向，小於 0 代表反方向，可過濾反方向的節點，以減少運算所需的時間，更快的建立逃生路徑。

3.5 智慧型行動裝置 GPS 定位

為了判別發生火災時使用者位於何處，需要製作一個 app 程式定位使用者的位置。利用 GPS 定位，使用 google map 顯示使用者所在位置。當使用者所在位置附近的大樓發生火災時，就會啟動本研究實作的模擬逃生程式，使用者即可觀看逃生路徑進行逃生。

四、遠端監控火災與逃生路線規劃系統實作

4.1 火焰偵測

本研究偵測火焰，不會使用整個影像，而是抓取有變動的區域，並且縮圖，以簡化其運算。使用 Sobel filter，對影像進行邊緣偵測；使用紋理化抽取紋理，將該影像物件的中間資訊凸顯出來，並將得到的兩個影像結合，輸出至下一個階段。

使用 YCbCr、RGB 與 HSV 將影像用顏色過濾，輸出火焰範圍，將影像中偏紅色、偏橘色的區域做交集的運算，當作火焰。將 RGB 的影像轉成 HSV，用顏色過濾範圍。將得到的影像作細線化，用骨架描述火焰面積，並做出骨架的節點、枝幹，確定是否為火焰。節點的情況如圖 2 火焰特徵。

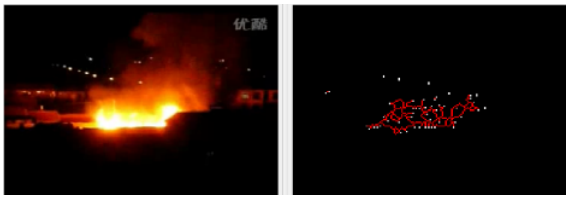

圖 2 火焰特徵

火焰有可能會因為風或其他因素影響而分散，搜尋該火焰附近的圖像，確定是否為一起或是分開的火焰。做完處理後，將圖像大小轉回原本的尺寸，並決定是否發布偵測到火焰的警報，程式的執行圖如圖 3 火焰偵測程式執行圖。

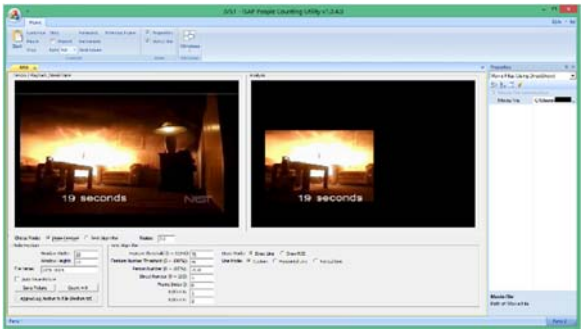

圖 3 火焰偵測程式執行圖

右方下拉式選項可選擇使用 IP Camera 偵測或是使用影片檔，圖為使用影片檔的方式去偵測，選定影片後按播放，即可進行火焰偵測。若確認有火焰，則會另外輸出火焰資訊檔案。

4.2 雲端平台建立

選用一台電腦當作伺服器端，並藉由火焰偵測程式進行監視，並將監視到的影像交由雲端平台進行運算，並將處理過的資訊儲存至伺服器端。

使用者透過具備連接網路的智慧型裝置，在任何地點與時間皆可取得雲端資訊，提高使用者的移動性與便利性。因此本研究撰寫程式，下載雲端平台上的火焰資訊，將火焰資訊提供給智慧型行動裝置上的程式進行運算。

4.3 建築物模型建立

利用 3D 建模程式 Sketch Up 3D 完成臺北科技大學的科研大樓，參考每層樓的平面圖，仔細規劃其教室位置、構造，並利用程式將其建出，讓使用者逃生時有同步的感覺。

整體架構分為一樓、二至十樓與十一至十六樓。一樓部分為獨立挑高，包含四個出口，格局構造也不盡相同，單獨出來實作。二至十樓包含半圓形大樓構造與矩形部分構造，兩結構合併建模。十一至十六為矩形構造，故獨立實作。

建構模型細節部分，首先利用線條工具，直線、圓弧等線條，加上形狀工具，矩形、圓形等形狀，繪製出各樓層平面圖。此處可利用架構分類的好處，二至十樓有相似的構造和內部格局，同步繪製平面圖，可將重複的構造先行繪製，最後再補上缺少的部分，或者構造上較特殊的部分，可大幅降低繪製時間，提高建模效率。同理可應用在十一至十六樓部分，至於一樓部分較為特殊則獨立繪製。

繪製格局為三個步驟中最為重要的一環，有誤差或者錯誤，都必須重新修改。在建立立體模型步驟中，依照正確的樓層平面圖配合推拉工具即可

完成，建築物每個樓層的高度或長度設定正確即可完工。

根據前面兩個步驟可得各樓層立體模型，接著使用環繞、平移、縮放等工具，再搭配鏡頭切換，俯視圖、底視圖、正視圖、後視圖、左視圖、右視圖等視角，將模型合併，即為一至十六樓的科研大樓模型。

為了簡化其工作內容，省略了教室內的桌子擺設，建立空無一物的教室樓梯，而樓梯則簡化成斜坡，以方便使用者可以上下樓梯，最終完成的科研大樓外觀如圖 4 科研大樓的建築物模型。

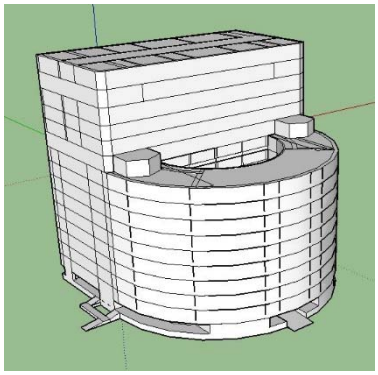

圖 4 科研大樓的建築物模型

4.4 逃生路徑實作

逃生路徑則是採用 A*路徑演算法，概念簡單且能迅速實作，又能即時產生建議之逃生路徑，符合本研究系統的需求。本研究使用 Unity 實作此系統。將之前已建立完畢的建築物模型匯入 Unity 3D，進行場地的建置。接著撰寫程式碼，建立 A*路徑演算法。

要使用 A*路徑演算法，就必須先定義節點(Node)、節點串列(Node List)與障礙物，用 Unity 3D 裡的絕對座標值作成 Node，讓系統可以去判斷範圍。在原本的 A*路徑演算法內加上向量的判定，在計算鄰居節點之前，利用起點至鄰居的向量與起點到終點的向量做向量內積，若小於零則將該鄰居節點加入 Negative Direction List，若大於零，代表同方向，則繼續判斷。使用目前節點至鄰居節點的向量與目前節點至終點的向量做向量內積，若小於零則將該鄰居節點加入 Reverse Move List，若大於零則計算該鄰居節點的評價分數。藉此判斷，可節省一些節點的計算，提高 A*路徑演算法的效率。

因為火焰可能會突然出現，本研究的逃生路徑設定為每秒重做一次，讓逃生路徑能夠即時的針對突然出現的火焰進行迴避。

發生火災時，使用者可能身處不同的教室、地點，因此本研究以按鈕選擇的方式，讓使用者先選擇樓層，再選擇教室，即可讓程式內的視角迅速移動到該教室，即可進行移動，選擇的方式如圖 5 選擇位置的程式執行圖。

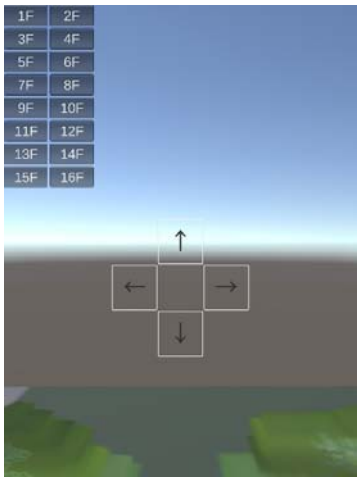

圖 5 選擇位置的程式執行圖

因為 Unity 3D 裡的建築物模型是 3D 的，本文為了提升 A*路徑演算法對 3D 模型的效能，加速並且簡化其運算，本研究根據使用者在 Unity 3D 內絕對座標值的 y 值，即為程式內高度，判斷使用者目前在哪一個樓層，並以此做為分界，一個樓層開啟相對應範圍的 A*路徑演算法，減少運算範圍，降低運算時間，讓使用者能依照建議逃生路徑迅速逃離現場。

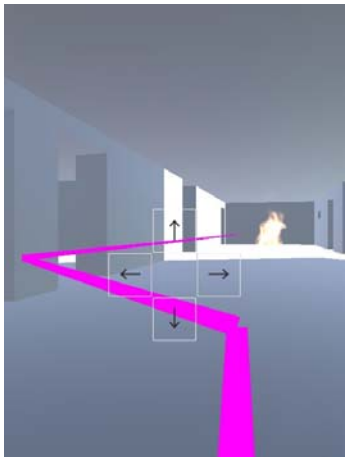

圖 6 程式繪出逃生路徑

科研大樓有兩側樓梯可進行逃生，因此逃生路徑的終點就有兩個。系統會先偵測出兩側的樓梯是否有火焰而無法通行，若有一側有火焰，則會無條件地將終點設為另一側並繪出逃生路徑。若兩側樓梯均無火焰，則系統會以使用者的位置為分界點，偵測使用者左半邊及右半邊的火焰數量，並將終點設在火焰數量較少的那一側。程式繪出逃生路徑的方式如圖 6 程式繪出逃生路徑。

科研大樓的 11 至 16 樓只有長方型的構造，但是 2 至 10 樓含有額外的半圓型構造，如 4.3 的圖 4 科研大樓的建築物模型，而系統的路徑偵測範圍不包含半圓形，需另外實作。本研究將偵測半圓形走道上是否有火焰，若都無火焰，則會指引使用者離較近的樓梯逃生。若走道上有火焰，則會指

引使用者往沒火焰的地方逃生。

本研究使用改良版A*路徑演算法，與原版A*路徑演算法相比較。在程式裡，從教室 229 的門口計算路徑至該層樓梯的入口，計算其所有在 List 節點內的數量，得到的實驗結果如表 2 改良版 A*與原版 A*之效能測試表。

表 2 改良版 A*與原版 A*之效能測試表

	改良版 A* 節點數量	原版的 A* 節點數量
OpenList	5	48
ClosedList	18	72
NegativeDirectionList	4	無
ReverseMoveList	78	無

依據改良版 A*路徑演算法，Negative Direction List 與 Reverse Move List 內的節點是沒有計算其 F 值，亦即為省下的節點運算數量。實驗結果顯示，改良版 A*只需要計算 23 個節點即能計算出路徑，而原版 A*卻需要計算 120 個節點才能計算出路徑。改良版 A*路徑演算法確實比原版 A*路徑演算法計算更少的節點數量，能較快的計算出路徑，使系統不會延遲。

4.5 智慧型行動裝置 GPS 定位實作

開啟該 app 程式時，就會用 google map 顯示使用者在何處。若 IP Camera 偵測到火災時，該 app 程式將會讀取雲端伺服器上的檔案，若讀取到火焰的資訊，就會發送 Intent，開啟 Unity 3D 匯出的 apk 程式，即可模擬火災逃生情形，定位 app 程式執行如圖 7 定位 app 程式執行圖。

圖 7 定位 app 程式執行圖

五、實驗結果

本研究使用 android 的智慧型手機及平板當作測試平台，做為進行火災發生時使用者手上的情況模擬。先將 GPS 定位的 app 程式與 Unity 3D 匯出的 apk 檔都安裝在智慧型行動裝置上，並開啟 GPS 定位的 app 程式先行定位。因為尚無架設 IP Camera，因此建立在雲端上的火焰偵測程式以監視影片的方式代替 IP Camera 的監視，火焰偵測程式監視到火焰時，就會建立火焰資訊的檔案，智慧型行動裝置上的定位程式讀取到火焰資訊檔案，並開啟模擬火災逃生情境的程式，使用者則可依據程式繪出的路徑開始進行模擬逃生。

5.1 可用性

從雲端平台自主的偵測火焰時間是平均是 0.02 秒，客戶端取得火焰資訊是 0.001 秒，演算法繪製好逃生路徑是 0.004 秒，共計 0.024 秒。經過 100 次的反覆驗證，平均流程完成時間是 0.02 秒。低於 2 秒，依據 NIST 所提供的聖誕樹火災影片，2 秒聖誕樹就完全燃燒，因此 2 秒內獲得資訊逃離現場是安全時間。

5.2 可靠性

火災現場網路可能不穩定，本研究即使斷網，逃生路徑演算法仍然可以運算，3D 建物仍舊存在。經過 100 次的反覆驗證，斷網地點不同，例如：樓梯間、走廊、教室、等。逃生路徑演算法及 3D 建物仍舊能於客戶端手機上運算，指示用戶逃離火災現場。

5.3 穩定性

雲端主機可承受多少客戶端連線，本研究模擬用戶端連線數達 1000 台裝置，並計算連線成功數與不成功數，成功定義為建立連線並在 2 秒內獲得火災資訊。不成功定義為建立連線但未在 2 秒內獲得火災資訊，或無法建立連線，即未獲得火災資訊。經過測試驗證，成功數為 999，不成功數為 1，穩定度為 99.9%。

六、結論

本研究設計與實作雲端監控火災與逃生路線規劃系統，在伺服器建立雲端系統，搭配 IP Camera 監控火災，並依據大樓的平面圖，製作該大樓的模型，套用改良版 A*路徑演算法，並偵測樓梯、走廊之間的火焰，即時規劃逃生路線並避開火焰，可讓使用者發生火災時，開啟本研究製作的 app 程式，即可立刻得知如何逃生，減少使用者會遇到的危險情況。

致謝

本研究由科技部計畫 MOST 104-2221-E-027-116 所補助，特此感謝。

參考文獻

[1] B. C. Arme, A. Ollero, J. R. Matinez de Dios, “An intelligent system for false alarm reduction in infrared forest-fire detection.”, IEEE Intelligent Systems and their Applications, vol. 15, pp. 64-73, 2000.

[2] L. Chu, S. Wu, “An Integrated Building Fire Evacuation System with RFID and Cloud Computing”, International Conference on Intelligent Information Hiding and Multimedia Signal Processing, vol. 7, pp. 17-20, 2011.

[3] T. Celik, H. Ozkaramanli, H. Demirel, “Fire pixel classification using fuzzy logic and statistical color model.”, IEEE International

Conference on Speech and Signal Processing, vol. 1, Honolulu, pp. I-1205-I-1208, 2007.
[4] S. Wang, D. Jeng, M. Tsai, “Early fire detection method in video for vessels”, Journal of Systems and Software, vol. 82, pp. 656-667, 2009.
[5] Maya, http://www.autodesk.com.tw/products/maya/features/all, Accessed May 3 2016.
[6] 3DS MAX, http://www.autodesk.com.tw/products/3ds-max/features/all , Accessed May 3 2016.
[7] A* Algorithm Brief, http://theory.stanford.edu/~amitp/GameProgramming/, Accessed May 3 2016.
[8] L. Chu, S. Wu, “A Real-time Decision Support with Cloud Computing Based Fire Evacuation System”, Nano, Information Technology and Reliability, NASNIT, vol. 15, Macao, pp. 45-48, 2011.
[9] L. Qiao, S. Zheng, Y. Wang, L. Zhang, “Application of Sketchup in the Teaching of Landscape Design of Undergraduate College”, International Conference on Optics, Photonics and Energy Engineering, vol. 2, pp. 306-308, 2010.
[10] 3D Warehouse, https://3dwarehouse.sketchup.com/, Accessed April 25 2016.
[11] Unity, http://unity3d.com/unity, Accessed May 6 2016.
[12] J. Ribeiro, J. E. Almedia, R. J. F. Rossetti, A. Coelho, A. L. Coelho, “Using Serious Games to Train Evacuation Behaviour”, Information Systems and Technologies, vol. 7, pp. 1-6, 2012.
[13] J. Hu, W. Wan, X. Yu, “A Pathfinding Algorithm in Real-time Strategy Game based on Unity3D”, Audio, Language and Image Processing, Shanghai, pp. 1159-1162, 2012.
[14] Z. Zhao, R. Liu, “A optimization of A* algorithm to make it close to human pathfinding behavior”, International Conference on Electrical, Computer Engineering and Electronics, vol. 2, Jinan, pp. 708-714, 2015.