以GPU加速SIFT演算法

蔡睿烝
國立臺灣師範大學
科技應用與人力資源發展學系
研究生
tempsteve1024@gmail.com

林政宏
國立臺灣師範大學
科技應用與人力資源發展學系
副教授
brucelinco@gmail.com

摘要

許多影像處理和電腦視覺的科技被應用於抓取圖片內容。其中 SIFT 廣泛地被各種應用方法採用，SIFT 最受歡迎的原因是其特徵描述不受尺度變換與旋轉變換影響的能力，以及不完全受到視角變換與明度變換的影響。近年來，影像處理單元因其對於大量資料強大的平行運算能力而受到相當多的注意。GPU 上的平行運算已被成功地使用於許多科學與工程應用。在本研究中，將透過 GPU 平行化加速 SIFT。以 linear search 來取代 KD-Tree，並且使用 GPU 平行化 linear search。期望透過原本在 CPU 環境上較為不便的 linear search，其高度可平行化優勢可以在 GPU 上展現更佳且更高精確度的效果。在實驗中發現，原本的 SIFT 演算法經過改良並使用 linear search 取代 KD-Tree 能夠加強精確度。而最後的實驗結果中顯示透過 CUDA 運算，演算法得到了將近六倍的加速。

關鍵字：尺度不變特徵轉換、圖形處理器、線性搜尋法、圖書管理、封面辨識

Abstract

Many image processing and computer vision technologies are being applied in image information extraction. Among of all, SIFT is extensively applied in plenty of applications. The reason that SIFT becomes more popular is because its feature descriptions are invariant to scale, rotation, illumination and viewpoint transformations. In recent years, GPU received considerable attentions with its powerful parallel processing ability. Parallel computing on GPUs has been applied in many science and engineering applications successfully. In this research, we intend to use GPU to accelerate SIFT algorithm. The program will use linear search method to replace KD-Tree algorithm, and GPUs to parallelism linear search method. We expect that linear search method will perform better result and higher accuracy because of its highly parallelizable advantage. The experimental results showed that after CUDA accelerating, the linear search method get a six-fold acceleration.

Keywords: SIFT, GPU, Linear search, Books management, Cover identification

1.緒論

尺度不變特徵轉換（Scale-Invariant Feature Transform, SIFT）為一種電腦影像辨識（image recognition）的演算法。SIFT演算法被廣泛地應用於許多領域，諸如物件辨識、3D建模、手勢偵測、影像追蹤等應用。其特點為假使一張圖片被更改其中的四項特徵，如「尺度（scale）」、「旋轉（rotation）」、「明度（illumination）」、「視點（viewpoint）」，SIFT依然能夠有效地辨識為

相同的結果（Sinha, 2000）。

進行SIFT演算法時主要有兩個步驟：「特徵點抓取（feature extraction）」與「特徵點比對（feature matching）」。特徵點抓取係經由六大步驟，以高斯模糊及梯度檢測等方法計算出圖形中的關鍵點後，建立特徵點集合供後段比對。特徵點比對的部分為一高度密集的計算過程，在傳統單執行緒（single-thread）的CPU架構中為相當費時的運算。

在SIFT演算法中，Lowe為了要降低整體程式的運算時間，在進行特徵點抓取的時，會根據些許規則省略許多特徵點來加快特徵點比對的進行，但是卻因特徵點省略而降低圖形比對的精確度。另在2004年的版本中，Lowe為了加速特徵點比對的效率，採用基於KD-Tree演算法延伸的BBF（Best-Bin-First）方法作為比對工具。此方法雖然可使程式的運算效率加快，卻存在著已知的「維數災難（curse of dimensionality）」問題，此問題會造成在高維度計算時讓比對的精確度降低。

圖形處理器運算（graphic processing units computing）已成為近幾年相當熱門的研究重點，本技術對於大量密集資料的運算有相當大的幫助。GPU運算亦由於此特點應用於流體動力學、分子動力學、地質探勘、大氣科學、資料探勘、金融財務等方面。初期的SIFT演算在進行特徵點比對的時候採用linear search方法，此方法將所有特徵點進行一對一計算的方式來歸納出比對的分組結果，對於單執行緒的CPU運算來說為一件耗時的工作，但此方法卻有著高度的可平行化優勢。於是以GPU的平行化運算來處理linear search會是一個不錯的解決方案。

基於以上論述，假設整體程式的運算時間相同，如果能夠加速SIFT特徵點比對的效率，使得後段的運算時間縮短，則在進行前段特徵點抓取的時候就可以不用為了節省運算時間而刪去太多的特徵點，以提升整體演算法的精確度。故本研究之目的為：

1.1 透過GPU運算架構加速linear search方法，加速SIFT後段的特徵點比對部分、增進整體程式效能。

1.2 提升整理比對精確度。

1.3 透過GPU加速過後，將來處理大量資料時可以進行多GPU的運算，達成海量資料處理。甚至在加速後期望能達到即時運算。

2.尺度不變特徵轉換

2.1 特徵點抓取

在SIFT演算法中，特徵點包含了許多圖形上的元素，並非只有尺度的差異。在人類的肉眼中，理解一張圖片中擁有的各項特徵是件容易的事。但是對於機器、電腦來說，要如何將這些看似「顯眼」、「特殊」的區塊化為重要的特徵值，SIFT演算法需要透過以下的步驟：

2.1.1 建構尺度空間（Constructing a scale space）：

在處理圖像的過程中，為了要找出那些顯眼的特徵，第一步就是要進行物件的模糊化。在SIFT中，程式會將圖像模糊後將大小縮為一半，這個步驟稱為一個「octave」，每個octave會產生五個圖像，象徵著這個圖像的「尺度」被增加了。此步驟是Lowe認為最佳的尺度變化，即四個octave、五次模糊。模糊化的過程為採用高斯模糊（Gaussian blur），其採用的公式如下：

$$\mathrm{L}(x,y,\sigma) = G(x,y,\sigma) * I(x,y)$$

$$\mathrm{G}(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

上列公式中，L為模糊化後的圖像、G為高斯模糊的參數、I為原始圖像。在本步驟中，程式將可以得到一系列高斯模糊處理的圖像，並在接下來的步驟中加以應用。

2.1.2 高斯拉普拉斯近似（LoG approximation）：

在前面一步當中，程式產生了一組逐漸模糊、收縮的圖像群，接下來要運用這些圖像產生另一組圖像，以下稱為「差異性高斯（Difference of Gaussians, DoG）」圖像群，這組圖像將會是找出關鍵點的重要圖像。一般的高斯拉普拉斯程序大致為將依圖形模糊化，接著計算其二階導數。經過這個程序的圖像，將可以被輕易的標示出圖中的邊緣或角。但是二階導數計算對於雜訊相當敏感，圖像中的雜訊容易造成程式的誤判，於是先前提到的模糊化將能有效地過濾雜訊，達到穩定計算之目的。二階導數計算量是非常密集的，對於程式執行的效率會有一定程度的影響，因此，原創者在SIFT演算法中對兩個連續的高斯模糊圖像做相減，如此一來可以快速地得出兩個圖片之間的差異圖像群DoG，取代原本需要密集運算的二階導數法，來快速的求出需要的圖像群，如：圖1所示：

2.1.3 尋找關鍵點（Finding key points）：

在前面一步中程式求出了高斯的差異值，而這一步演算法將會尋找這些值當中的最大值與最小值，並且將他們標示出來。在本步驟中包含兩個部分：「在DoG圖像中尋找最大值、最小值」以及「尋找子像素的最大值、最小值」。

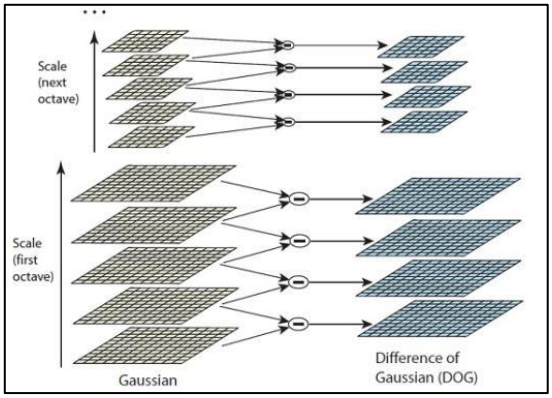


圖1 DoG圖像群產生示意圖

資料來源：出自Sinha（2000）。

在第一個部分中，程式將經歷每一個像素，並且檢查所有鄰近它的像素，這樣的像素將會有26個，如圖2所示。X代表了正被檢查的像素，在其立體空間相鄰的所有像素點便是接著要被檢查的範圍。但有許多的像素點在檢查完26個鄰居之前就會先被程式判定無用而遺棄，只有真正有價值的像素會被檢查完。

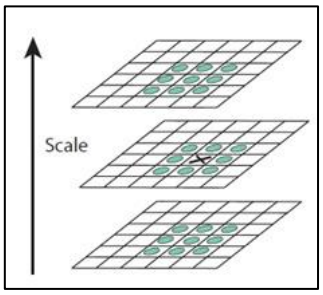


圖2 像素探訪示意圖

資料來源：來自Sinha (2000)。

2.1.4 去除不佳的關鍵點（Get rid of bad keypoints）：

為了消除那些對於比對沒有幫助且耗費

時間的關鍵點。首先進行的是消除對比度較低的關鍵點，程式會訂定一個臨界值，當一個關鍵點的DoG值之絕對值小於該臨界值時，這個關鍵點將會被視為無效的，並予以刪除。接著進行「邊緣」的刪除，程式給定關鍵點兩個垂直的梯度（gradient），並且判斷兩個梯度的值。當兩個梯度都較小時，視為平面；一大一小則為邊緣；兩個梯度都大時則可定義為角。角對於圖像辨識來說是一個非常重要的特徵，所以這些點將不會被忽略。

2.1.5 決定關鍵點的方向性（Assigning an orientation to the key points）：

方向性定位的目的便是為了「旋轉的不變性」，首先要收集所有關鍵點的梯度方向以及大小，程式會依這些關鍵點的「尺度」來決定需要的收集區大小。在這塊收集區中，程式會找出這個區域中最突出的方向來指定為關鍵點的方向，如圖3所示：

2.1.6 產生SIFT特徵（Generate SIFT features）：

在一連串的轉換之後，程式會將每個關鍵點轉換成為一個特殊的「指紋（fingerprints）」，此指紋將能夠將這些關鍵點表示為獨一無二的特徵點，並且包含了128個數值，相當於這關鍵點周圍的16個像素區塊的8個向量定義值。

對於兩張圖片的SIFT特徵比對來說，關鍵點的比對就是透過最近鄰居法來比較其歐式距離。圖4示範了如何用SIFT來比對兩張圖片。在圖4（A）中，關鍵點會指示位置、尺度與方向性的向量。左邊的圖擁有1982個特徵點、右圖有3956個特徵點。在圖4（B）中將可以找到1971條正確比對的連接線。

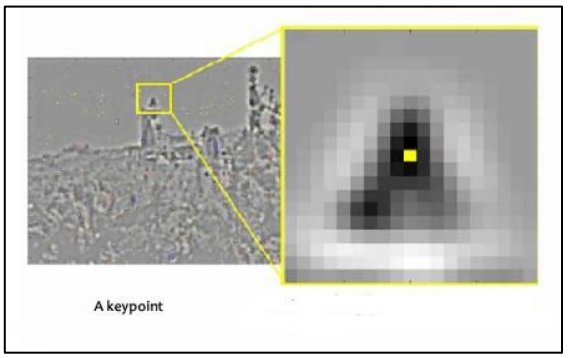
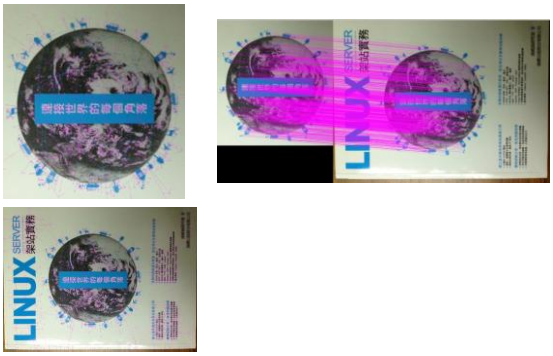


圖3 定位關鍵點方向

資料來源：出自Sinha (2000)。



(A) 特徵點　　(B) 特徵點比對

圖 4 SIFT 比對範例

2.2 近鄰搜尋法

k-近鄰搜尋法（k Nearest Neighbor Search Algorithm）為一種慣用的機器學習（Machine Learning）演算法。機器學習目的是讓機器接收外界輸入的資料之後，依照某種演算法訓練出一種模型（model），有了模型，機器看到新的資料的時候，就能夠有某種程度的經驗與智慧來對新的資料產生理解。機器學習分為「監督式」與「非監督式」，主要以是否需要訓練階段而定義，而k-近鄰演算法則屬於「監督式」的機器學習法。

k-近鄰演算法顧名思義，便是去尋找距離查詢點最近的k個鄰居，以分類查詢點為

例，當機器接收到一個查詢點（圖5中的綠色圓點）時，機器會依照某些值將查詢資料去與散布在一空間中的所有參照點（圖5中的藍色方點與紅色三角點，分別代表兩個類別）去做相似度或是距離的計算。當使用者決定一個k值之後，機器會尋找出與查詢點最相似或最接近的k個點，並且將這k個點做整理，如果這些點中有某個類別的比例比較高，則機器會將該查詢點歸類在比例較高的類別當中。如圖5所示，假如以k=3作為依據，可得到實線圓圈的範圍，由於紅色的三角形點數目較多，則查詢點可以被歸類為紅色三角點；若以k=5為依據，可得到虛線圓圈的範圍，則因藍色方點的數量較多而會將查詢點歸類在藍色方點的類別中。

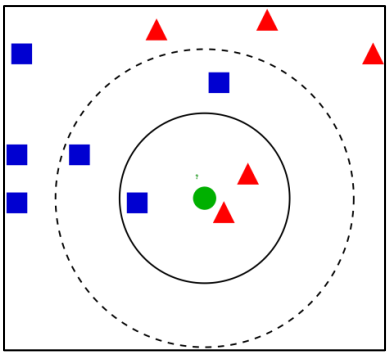


圖5 k-近鄰搜尋法示意圖

資料來源：出自Tarhini(2012)。

根據以上提到的演算法步驟，在程式當中呈現的方式，最直接的做法就是一對一比對兩份資料所有值的差（亦稱線性搜尋法，linear search algorithm），所以如果資料一有M筆資料、資料二有N筆資料時，則必須要做M*N次計算，隨著資料量越來越多，程式的執行效率也會越來越慢。

由於這項缺點，Lowe在2004年提出的SIFT版本中採用了基於K-維樹搜尋法（k-dimension tree algorithm）延伸的BBF演算法。K-維樹在1975年由Bentley發明，是一種二元樹，其每個節點（node）皆為一個k維度，即將上述的k-近鄰搜尋法中的各個參照點先整理進一個二元樹中，並將可用的數值存在每一個節點中，如此一來，當查詢點被機器接收到的時候，機器可以快速的利用二元樹搜尋方式得到解答，這樣的運算方式在CPU的運作方式中可以比linear search法快。但由於「維數災難」的影響，K-維樹的方法計算出來的結果會有誤差，精確度較低。

2.3 綜合分析

本節介紹的幾種演算法，皆有後續學者研究將其延伸、改進，如同SIFT演算法就比對方式改進為BBF演算法。但是猶如前一段所述，K-維樹因為會出現誤判的情況，加上其可平行化的程度並不高，所以將linear search法做平行處理，透過其高精確度與高度可平行化的特點，達到為整體程式加速為本研究之想法。

在先前學者研究方面。在CPU運算階段，Bosch、Zisserman、Muñoz等人（2008）與Jégou、Douze、Schmid等人（2011）還有Liu、Yuen、Torralba等人（2011）的文章中皆有提出以近鄰搜尋法加速SIFT演算法的想法並加以實驗。另外，Bonato、Marques、Constantinides等人（2008）、Huang、Huang、Ker、Chen等人（2012）以及Huo、Pan、Huo、Zhou等人（2012）的文章則提出以平行化的演算法進行SIFT演算法的加速，無論是使用硬體加速亦或是軟體加速皆可以獲得不錯的實驗結果。

而在GPU演算發明之後，Heymann、Muller、Smolic、Frohlich、Wiegand等人（2007）提出透過圖形處理器平行化語言

GPGPU 對 SIFT 演算法進行加速。其後在 CUDA 被發表之後，Warn、Emeneker、Cothren、Apon 等人（2009）也提出使用 OpenMP與CUDA進行SIFT的加速。兩者皆對於原始的SIFT版本得到不錯的加速效果。就本文欲加以改進使用的k-近鄰搜尋法而言，Garcia、Debreuve 在 2008 年以及兩人與 Nielsen、Barlaud於2010年發表關於使用GPU架構加速k-近鄰搜尋法的兩篇文章。本研究除了著重在暴力近鄰搜尋法的平行化處理之外，將會把改進後的演算法結合SIFT演算法，來增進整體SIFT演算法的效率。並且在之後的研究中，加入多 GPU 運算（multi-GPU）方法以及將前端的SIFT特徵點抓取也改為採用GPU運算。

3.技術應用

CBIR（Content-Based Image retrieval，基於內容的圖像檢索）為一種使用圖像中的實際內容而非關鍵字、標籤或圖片描述，像是色彩、形狀和紋理等來從一個大資料庫中檢索數位圖像的電腦視覺科技應用程式。CBIR被應用於許多方面，如建築與工程設計、藝品收集、犯罪防治、地理資訊遙測系統、智慧財產、醫療診斷、軍事、照片檔案與零售目錄等。

圖6呈現一個基於SIFT的CBIR系統。樣本圖片將被轉換為SIFT特徵的方式存於資料庫中。當得到一個檢索圖片，圖片的SIFT特徵將會被抓取並與資料庫中的資料做歐式距離（Euclidean-distance）比對。與檢索圖片歐式距離最相近的圖片將會被指定成比對成功。而隨著資料庫大量的增加，最近鄰居（nearest neighbor）的比對過程將會成為CBIR系統中計算最密集的程序。

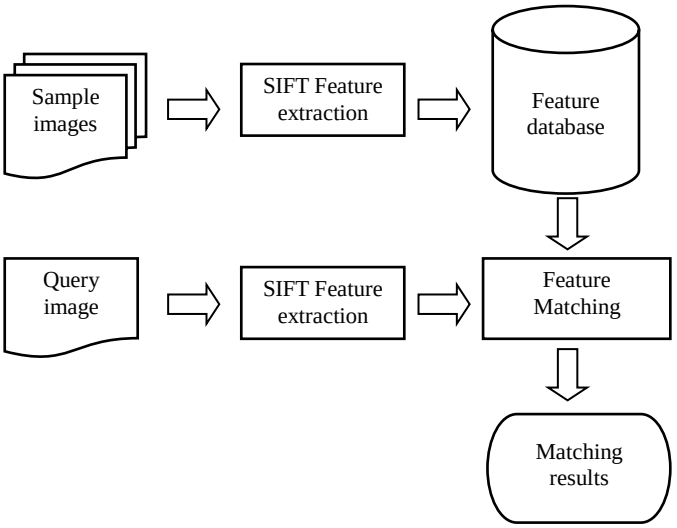


圖 6 基於 SIFT 的 CBIR 系統

本研究計畫呈現一個使用圖形辨識技術來將書本封面做整理的系統。系統藉由 SIFT 的圖像辨識功能，將圖書的封面轉為特徵值，並且建立特徵值資料庫。當使用者想要查詢一本書的時候，只需將欲查詢圖書之封面輸入資料庫，系統便可以根據特徵比對的結果輸出，在資料庫中找尋到查詢的圖書資訊。在系統的建置階段，必須要將館內現有的藏書做建檔工作，建檔為採用經 CUDA 技術改良之 SIFT 演算法，將圖書之封面使用數位攝影機拍攝後，透過演算法轉換為特徵點集合，集合中的每一個特徵點都擁有 128 個特徵值可供比對。建置完成後，系統會得到一個全館藏書封面的特徵點資料庫，每一筆資料都會對應書籍資料庫。當使用者欲使用本系統查詢藏書資料時，只需將封面拍下並上傳至系統，系統會將該封面轉換為特徵點，並使用 GPU 做平行比對出待查的書籍。

4.平行化運算架構

4.1 CPU上的平行運算

本研究將先使用OpenMP套件進行CPU架構上的初步平行化，以評估整體實驗的可行

性。一般的程式執行時只會用到一個執行緒，所以當程式在進行迴圈處理的時候，控制項有幾個就必須要一個一個跑到結束為止，更遑論在暴力近鄰搜尋法中必須要用到雙層迴圈。採用OpenMP以後，程式可以在被指定到的區塊使用多個執行緒來同時運算，以Intel® Core™ i7-3770處理器為例，在Intel® Hyper-Thread™的架構中擁有8個可執行緒核心，所以這個部分將能夠同時有8個執行緒進行運算。假設迴圈中的每個步驟所耗費的時間相同，則理論上整體迴圈運算時間將會縮短為原本的八分之一。

4.2 GPU上的平行運算

如緒論所述，GPU運算由於其對大量密集運算的平行化優勢，在近幾年受到相當熱烈的關注。而其中CUDA（Compute Unified Device Architecture，統一運算架構）更是GPU運算平台的指標性解決方案。在本實驗中，我們會將特徵點抓取所得到「指紋」中的128個特徵值儲存到GPU中的一個thread中，並且在開啟兩圖特徵點數相乘大小的block數目。在CUDA進行運算的時候，同時就會有128個thread在兩個block中進行計算。以NVIDIA® GeForce® GTX680為例，該顯示卡的GPU晶片擁有1,536個核心，於是同時在理論上可以執行1,536個執行緒，與CPU上面的8個執行緒相比起來有相當大的差距。以下圖為例，當有一圖片的特徵數為N個時，其在顯示卡記憶體中的儲存方式如圖7所示：

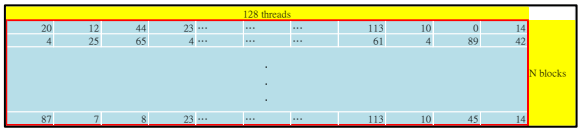


圖7 GPU儲存示意圖

資料來源：自製。

5.實驗結果

5.1 實驗介紹

本實驗有兩個要收集的主要目標「準確度」以及「運算時間」。收集比對準確度的主要目的是要證明 Linear search 法與 K-D Tree 之間準確度的差異，如前面所述，K-D Tree 會因為維數災難的影響，導致在比對的過程中出現過多錯誤的連線組合（mismatch）。於是透過比較 mismatch 出現的次數，來歸論出哪種比對演算法較佳。在收集運算時間的方面，便是比較程式運算所需要的時間長短，由於本研究尚未對特徵抓取的部分作加速，所以大部分的程式所花費的時間都是在特徵點抓取上，但猶如緒論所述，若本研究可以將特徵點比對的運算時間也做到縮短，未來在將特徵點抓取的步驟也做加速之後，便能達到降低整體運算時間的功效。

本實驗環境包含 Host 端以及 Device 端。Host 端使用 Intel® CoreTM i7-3770 的 CPU、16GB DDR3 記憶體和 Linux x86_64 作業系統。Device 端使用 NVIDIA® GeForce® GTX 680 的 GPU，其擁有 2,048 MB 的 GDDR5 記憶體，並使用 CUDA 5.0 toolkits。

5.2 精確度實驗

在精確度實驗的部分，本研究會透過兩種圖形方面的轉換來比較 Linear search 與 KD-

Tree 方法的 mismatch 出現次數。

第一種是尺度，實驗組為一張圖片之原始尺寸與縮小成 50%、25%、12.5%之後的縮小圖，如下表 1 所示。在尺度實驗中，我們會比較 linear search 與 KD-Tree 兩種方法在不同的尺度比較中產生的 mismatch 數目，結果如圖 8。從圖 8 中可以發現 KD-Tree 方法在尺度越來越小的比較中發生越來越多的 mismatch，顯示在尺度有變化的情況之下，KD-Tree 的方法會產生不精確的結果。

表 1 尺度實驗

Linear search	KD-Tree
	

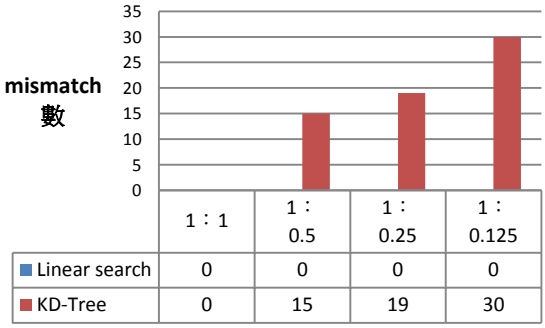


圖 8 尺度實驗結果

第二項實驗是由旋轉不變性的方面來進行比較的實驗，實驗組為與原圖經過順時針旋轉 90°、180°、270°的圖片，如下表 2 所示。同樣的，我們會針對兩個方法在實驗過程中產生的 mismatch 數目來驗證其比對的精確度。而根據圖 9 的實驗結果，可以發現 KD-Tree 方法的 mismatch 數目也會因為原圖經過旋轉之後而產生，linear search 的方法雖然在 180°的實驗中出現一條 mismatch，但是整體表現仍較 KD-Tree 來得良好，足以顯示 linear search 在精確度上面優於 KD-Tree 方法。

表 2 旋轉性實驗

Linear search	KD-Tree
	

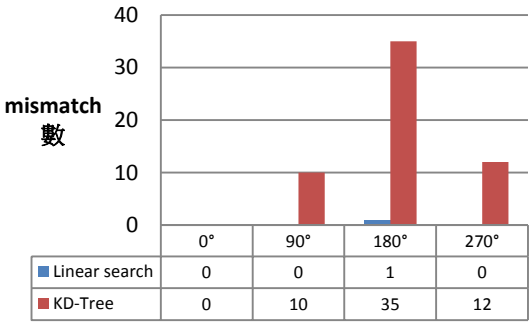


圖 9 旋轉性實驗結果

5.3 加速實驗

目前的加速效果方面，我們先進行特徵點比對部分的 GPU 加速，本實驗將輸入兩個特徵點接近 20,000 個的圖片（接近本組可用 GPU 之記憶體極限），並且與 CPU 運算組、OpenMP 加速組比較時間。

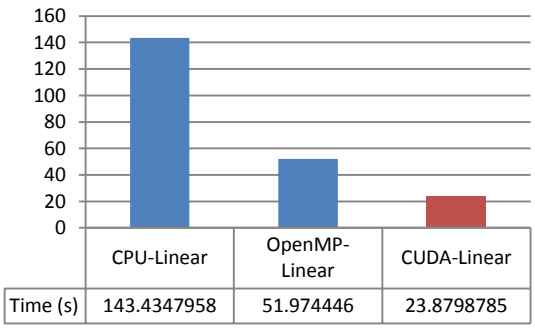


圖 10 程式執行時間

由圖 10 可以發現，使用 CUDA 加速之 linear search 相對於在 CPU 上的結果已可以大幅減少運算時間。由於前面的兩個結果顯示出 linear search 在精確度上的優異性，對於該方法的加速以及改進將會是非常重要且會立即著手進行的工作。

6.結論

總結以上，本研究提出一個基於改良過後的 SIFT 演算法而製作的 CBIR 系統。原本的 SIFT 演算法經過改良也使用了 linear search 取代 KD-Tree 而加強了精確度，並且透過 CUDA 得到了將近六倍的加速。期望可以透過書本上原始的資訊，就可以達成檢索書庫的功能。

從文獻中可以觀察出，原本的 SIFT 演算法確實有相當大的加速空間。而無論是透過改良特徵點抓取或是特徵點比對的演算法，在先前的研究中並沒有進一步將改良後的演算法使用在平行運算架構中。而提出平行運算的文獻也僅限於進行原本演算法的平行化，並未如同本研究將特徵點比對部分的方法使用 Linear Search 法。根據本研究的實驗結果，將特徵點比對的演算法改為 Linear Search 法後，相較於原演算法不僅可以提升比對精確度，亦可以增加運算速度。

本研究認為，未來的研究可進行程式最佳化、並且將前段的特徵點抓取進行 GPU 加速。另外，在實驗中發現本程式可以容許的單一圖片最大特徵點數量約莫是 20,000 個特徵點左右，其原因為顯示卡記憶體的受限。未來之研究可嘗試將超過單一顯示卡負荷的資料透過多重顯示卡（multi-GPU）的方式處理，以運算過大的圖片或是特徵點過多的圖片。

參考文獻

[1] Bentley, J. L., Multidimensional binary search trees used for associative searching. ***Communications of the ACM***, Vol. 18, No. 9, pp. 509-517, 1975.

[2] Bonato, V., Marques, E. & Constantinides, G. A., A parallel hardware architecture for scale and rotation invariant feature detection. ***IEEE Transactions on Circuits and Systems for Video Technology***, Vol. 18, No.

12, pp. 1703-1712, 2008.

[3] Bosch, A., Zisserman, A. & Muñoz, X., Scene classification using a hybrid generative/discriminative approach. ***IEEE Transactions on Pattern Analysis and Machine Intelligence***, Vol. 30, No. 4, pp. 712-727, 2008.

[4] Garcia, V., & Debreuve, E., ***Fast k nearest neighbor search using GPU***. Paper presented at the 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, Anchorage, Alaska, USA, 2008.

[5] Garcia, V., Debreuve, E., Nielsen, F., & Barlaud, M., ***K-nearest neighbor search: fast GPU-based implementations and application to high-dimensional feature matching***. Paper presented at the International Conference on Image Processing (ICIP), Hong Kong, 2010.

[6] Heymann, S., Muller, K., Smolic, A., Frohlich, B., & Wiegand, T., ***Sift implementation and optimization for general-purpose gpu***. Paper presented at the 15-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, Bory, Czech Republic, 2007.

[7] Huang, F. C., Huang, S. Y., Ker, J. W. & Chen, Y. C., High-performance SIFT hardware acceleration for real-time image feature extraction. ***IEEE Transactions on Circuits and Systems for Video Technology***, Vol. 22, No. 3, pp. 340-351, 2012.

[8] Huo, C., Pan, C., Huo, L. & Zhou, Z., Multilevel SIFT matching for large-size VHR image registration. ***IEEE Geoscience and Remote Sensing Letters***, Vol. 9, No. 2, pp. 171-175, 2012.

[9] Jégou, H., Douze, M. & Schmid, C., Product quantization for nearest neighbor search. ***IEEE Transactions on Pattern Analysis and Machine Intelligence***, Vol. 33, No. 1, pp. 117-128, 2011.

[10] Liu, C., Yuen, J. & Torralba, A., SIFT flow: Dense correspondence across scenes and its applications. ***IEEE Transactions on Pattern Analysis and Machine Intelligence***, Vol. 33, No. 5, pp. 978-994, 2011.

[11] Lowe, D. G., Distinctive image features from scale-invariant keypoints. ***International Journal of Computer Vision***, Vol. 60, No. 2, pp. 91, 2004.

[12] Sinha, U., ***SIFT: scale invariant feature transform***. Retrieved October 5, 2012, from http://www.aishack.in/2010/05/sift-scale-invariant-feature-transform/, 2010.

[13] Tarhini, A., ***Parallel k-nearest neighbor***. Retrieved October 10, 2012, from http://alitarhini.wordpress.com/2011/02/26/parallel-k-nearest-neighbor/, 2011.