中華民國第 59 屆中小學科學展覽會

作品說明書

高級中等學校組　電腦與資訊學科

第三名

052503

提升多張 QR Code 條碼辨識之研究

學校名稱：桃園市立內壢高級中等學校

作者：	指導老師：
高三　黃子軒	詹國煇

關鍵詞：QR Code、Interpolation Filtering、Resampling

摘要

QR Code 的應用在我們的生活當中隨處可見。然而，引起我們好奇的是，為什麼手機的 QR Code APP 都只能辨識一個 QR Code？這對於要辨識大量 QR Code 的人，就顯得過於不方便。因此，我們在想能不能有一次辨識幾十個，或甚至上百個 QR Code 的方法？

本研究針對目前辨識率最高的開源 QR Code 辨識函式庫 Zbar library 作改進，發現使用 up sampling 搭配多種不同 Interpolation Filtering，可以有效提升多個 QR Code 的辨識率。如果再加上本研究提出的 module sampling 與 match centers 方法，這樣一張相片中包含 112 個 QR code 的辨識率可以提升到 93.3%，而平均 QR Code 辨識時間也縮短為原本的 36%。

這個研究成果，希望未來能回饋到 Zbar open source library，讓手機 APP 的開法者可以運用，一般使用者可以享受到小小手機也可以準確辨識多張 QR code 的便利性。

目前現行的做法

本研究改進的方法

壹、研究動機

在近代的統一發票中，因為二維條碼資料容量大、可附加 logo 標商效果且具有容錯機制，所以使用方面逐漸取代了一維條碼。而這些發票經由一般手機應用程式掃描後，就能夠確認其中的資料訊息，無論是在日期的檢視、號碼的比對及購物清單資料管理上，均可以加速作業的進行。然而，對於一些需用掃描大量條碼的使用者來說，如果一張張掃描的話，將耗費龐大的時間。以慈善基金會為例，近年來，每年的國人捐獻發票量都是以千萬張為單位，要在短時間內利用人力或一般手機的應用程式完成比對是極耗時間的。綜合以上，本研究希望能夠設計一套「同時處理多張統一發票」的應用程式，來解決以上的問題。

貳、研究目的

現今國內外大部分 QR Code 的辨識 APP，底層大多是利用兩套開源 (open source) 的 QR Code 辨識函式庫 (library)。第一套系統網路上首推 ZXing library，拿來辨識單張 QR Code 都沒問題，但是一張相片如果包含不超過 10 個的 QR Code，發現幾乎都是失敗，連一個 QR Code 都辨識不出來。第二套系統是 Zbar library，這套系統比較好，上百張 QR Code 的辨識率約在 44.6%之間，但是離真正實用，還是有一段差距。本研究希望可以透過改善現有的函式庫，將其辨識率提高、縮短辨識所需的時間，發展出更為實用且完整的統一發票辨識系統，提供於社會使用。

參、研究設備及器材

一、 硬體

（一）任何具有瀏覽器(browser)的電腦（撰寫程式）

（二）Google Cloud Platform 的 VM instance（撰寫與執行程式）

二、 軟體及工具

（一）C / C++

（二）ZBar library

（三）ZXing library

（四）統一發票 QR Code 條碼

（五）照相手機

肆、研究過程或方法

一、財政部的電子發票規範

電子發票上的條碼本身主要用來儲存發票字軌號碼與記錄訊息資料，總共有三個條碼，其同時包含了一維條碼和二維條碼。依財政部規定，二維條碼應以 QR Code 規格，數量為兩個：

（一）左方二維條碼記錄

該紀錄包含發票字軌號碼、日期、隨機碼、金額、完整品目比數與驗證資訊等。

（二）右方二維條碼記錄

以 ** 為起始符號，用來判辨左右方之分，接續左方不敷記錄，將剩餘之資料記錄於右方。

綜合以上，一維條碼只能儲存英文數字和標點符號，二維條碼則可儲存不同編碼與多媒體的資訊。二維條碼可以儲存更多的資訊量，又帶有容錯的功能，所以在應用上，逐漸超越了傳統的一維條碼。

二、QR Code 統一發票條碼

二維條碼改良了一維條碼一些不足之處，例如資料容量大、容錯性高等。因此，近年來二維條碼的使用上漸漸取代了一維條碼。常見的一些二維條碼有層排式的 PDF417、Code49、Code16K 等，矩陣式的 Maxicode、Data Matrix、QR Code 等。QR Code（Quick Response Code）即為矩陣圖碼的一種。

QR Code 圖碼成四方形，獲得 IOS 國際標準採用。其分別由以下部分所成：

（一）Finder patterns：由此可快速找到 QR Code 的位置。
（二）Alignment Patterns：可以提高 QR Code 的識別率。
（三）module：QR Code 中最小的基本單位，由黑白碼元所組成（黑色表示 1;白色表示 0），由此來組成真正的數據部分。
（四）Error Correction：進行一些錯誤的修正。

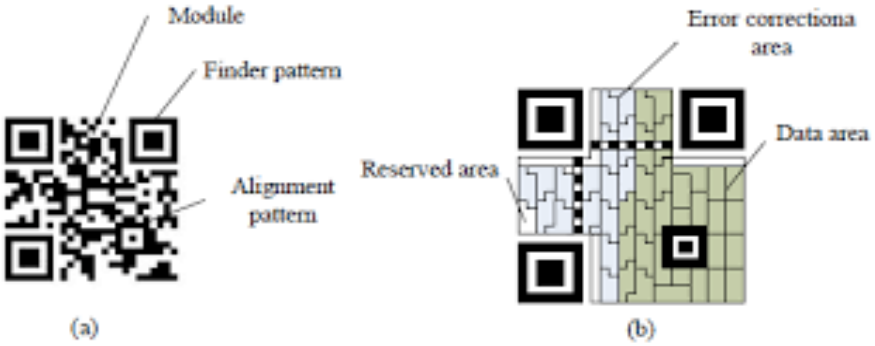


圖 1: QR Code 條碼結構 [截自網路上_淺談 QR Code]

QR Code 碼共有 40 種版本，每增加一個版本，常與寬個加 4 個 module，每個版本的最大資料容量也會隨著版本增加而擴大。Version 1 為 21 x 21 modules，Version 40：177 x 177 modules。參考財政部規定，左方二維條碼應使用 QR Code V6(41x41)(含)以上版本，並採用 Level L(容錯率 7%)以上之防錯標準。

三、Reed－Solomon 介紹

Reed－Solomon 是個非常重要的錯誤更正碼，廣泛的運用在各個通訊上。QR Code 中也有 Reed－Solomon 的功能，其運作功能為修復字符的破壞。主要分成四種容錯等級，分別為 L、M、Q、H，不同的等級，可接受的損毀率也更高。適用的等級也會隨著資料含量有所調整。

QR Code Error Correction Capability*	
Level L	Approx 7%
Level M	Approx 15%
Level Q	Approx 25%
Level H	Approx 30%

圖 2: [截自網路上_Error Correction Feature]

在二維條碼中，字符的損壞是無可避免的，尤其是越長的字詞越容易出現破損、分離。為了避免這類的情形發生，QR Code 也運用上了 Reed－Solomon 來保持資料的完整性，以下用例子來說明：

THIS
THAT
CORN

圖中有幾個英文字母 this、that 跟 corn，當資料被損毀而掃描出來的為 co** 時，從備份的字典中能明確的找出此字詞即為 corn。然而，當掃描出來的資料為 th**時，備份字典中會有重複字母的問題，而無法恢復完整的訊息。解決此問題的最簡單方式，就把每個單字後面加長，讓每個單字都是獨特的。

THISABCD
THATBCDE
CORNCDEF

圖中在單字後面都進行增長，使 this 與 that 可以運用多餘字元來恢復原始數據。前面範例粗略的顯示 Reed-Solomon 如何工作。Reed-Solomon 還會進一步的將冗餘數據傳送到 Galois Finite Field 數學演算法中。

四、Truncation error

Truncation error 為 Finder Patterns 的格式判斷，高解析度下才能做的辨識。Finder Patterns 一般的格式比例為 1：1：3：1：1，當判斷的比例錯誤或像素不足導致辨識不清時，就會出現 Truncation error，比例格式如下圖：

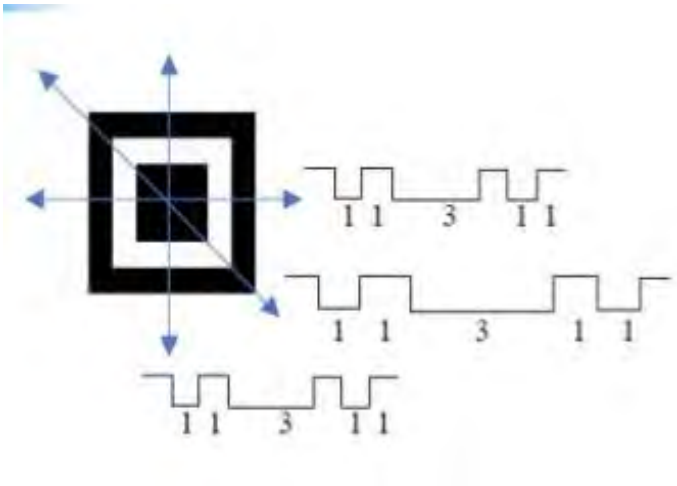


圖 3: [截自網路上_淺談 QR Code]

五、Resampling and Interpolatoin Filter

Resampling 是一種增加和減少像素（pixel）的工具，其中含有 up sampling 和 down sampling。而 Interpolation Filter 是一種新的頻率合成技術，能將新增出來的像素進行內插取樣。在實驗中，因為一張相片含有 100 多個 QR Code，解析度大幅降低，所以每個 QR Code 所佔的 pixel 非常少。為了提高辨識率，用到了 up sampling 來增加每張相片的 pixel。有了足夠的 pixel 後，再利用 Interpolation Filter 內插的方式取樣。

以下介紹幾種常見的 Interpolation Filter：

（一）Box & Point

Box Filter 與 Point Filter 的處理方式相似。在條碼影像處理中，表現是比較普通的，比較適合運用在其他處理工作上。

（二）Nearest

Nearest 是其中一種構造簡單的 Filter，其內插方式主要是給每個鄰居權重賦值為 1/d，其中 d 為到鄰居的距離。原本我們預測它會是所有 Filter 中成效最好的。結果實驗數據出來後，不僅辨識率為低迷的 34%，就連從圖形中也能名顯看出圖形解析度的差距。

圖 4: [截自網路上_Cornell Computer Science]

如圖，同樣都是將 10 倍小的原圖 up sampling，左邊為 Nearest Filter 所處理過的圖案，與右邊相比，僅用肉眼就能辨識出圖形的優劣。

（三）Lanczos

主要被當做是低通過濾器(low pass filter)，或是數對訊號 resampling 的內插法，普遍被認為是簡單 filters 裡的“best compromise”。

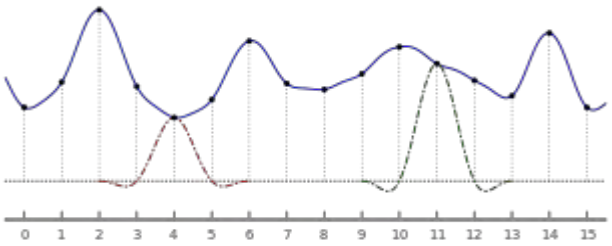


圖 5 [截自網路上_維基百科]

黑色點為原先訊號，紅色與綠色為 Lanczos kernel，藍色為經過 Interpolation 後的平滑曲線。

六、Barcode open source library

（一）ZXing

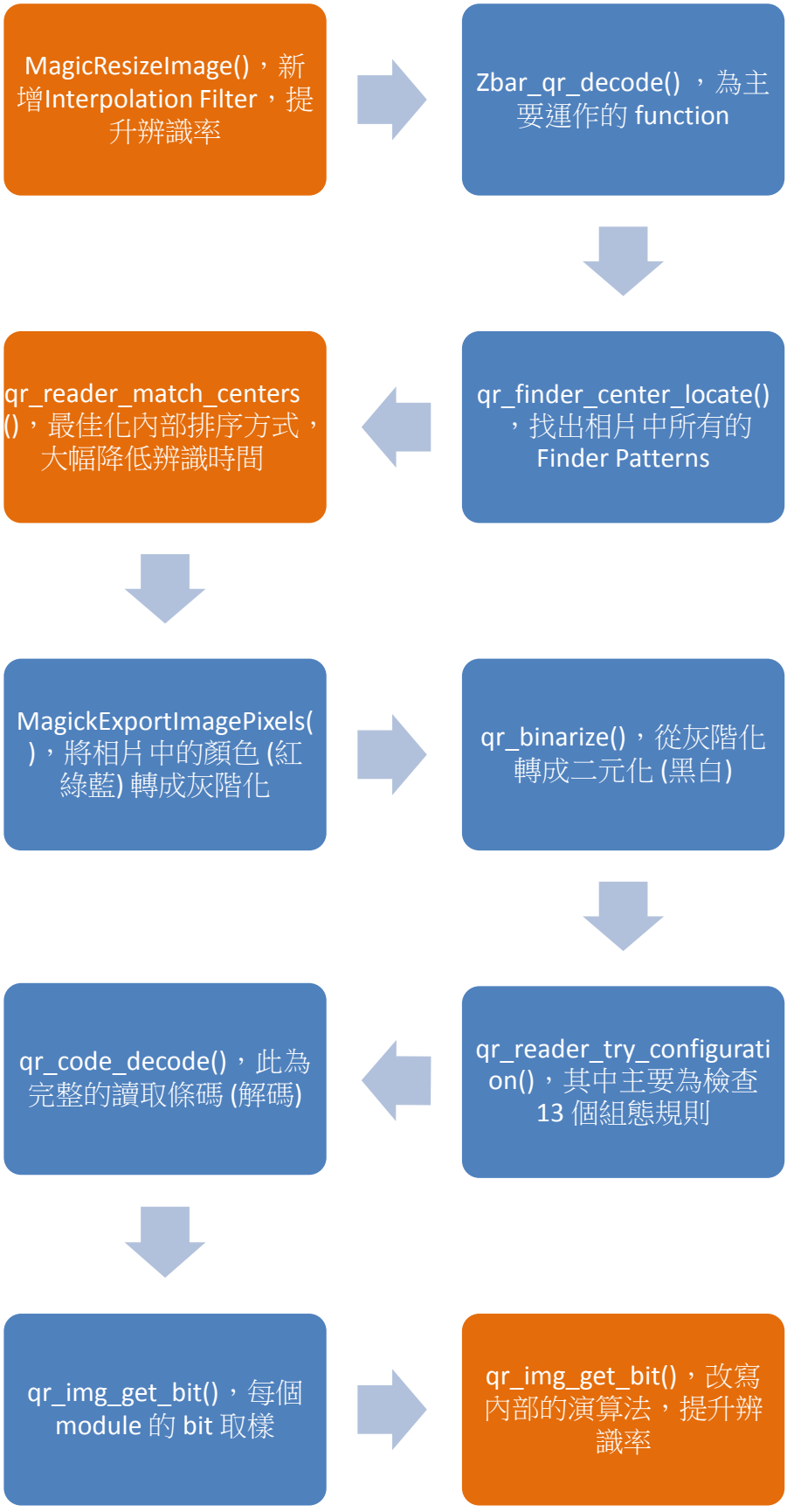
是一種用 Java 語言所撰寫的開源軟題套件，是目前最流行的 QR Code 解碼套件之一。不過，系統安裝與設定較為複雜，不適合初學者使用。而且，本研究的實測顯示，此系統僅適合應用在單一 QR Code，對於多張 QR Code 的辨識幾乎完全失敗。

（二）ZBar

是一種用 C 語言所撰寫的開源軟體套件。函式庫中有許多方便的函式，可以靈活的應用在有關條碼掃描與辨識解碼等相關工具上，圖形掃描的應用程式，安裝設定簡便，對於多張 QR Code 有最高的辨識率，為本研究之主要實驗與改進對象。

七、改進 QR Code 辨識流程構想

以下為 Zbar 運作的流程圖，條碼辨識的主要過程分成以下幾種分述。橘色為改進演算法，目的為提升辨識率所提出的步驟。


MagicResizeImage()，新增Interpolation Filter，提升辨識率
Zbar_qr_decode()，為主要運作的 function
qr_finder_center_locate()，找出相片中所有的 Finder Patterns
qr_reader_match_centers()，最佳化內部排序方式，大幅降低辨識時間
MagickExportImagePixels()，將相片中的顏色 (紅綠藍) 轉成灰階化
qr_binarize()，從灰階化轉成二元化 (黑白)
qr_reader_try_configuration()，其中主要為檢查 13 個組態規則
qr_code_decode()，此為完整的讀取條碼 (解碼)
qr_img_get_bit()，每個 module 的 bit 取樣
qr_img_get_bit()，改寫內部的演算法，提升辨識率

：為本研究改善之模組

1.使用 Interpolation Filter

一張相片內用了 56 張電子發票，其中包含了 112 個 QR Code（一張發票有 2 個 QR Code）。雖然發票中也有一維條碼可以掃描，但因為一維條碼解析度太低，再加上沒有容錯機制，所以在掃描的過程中條碼辨識不出來。因此一維條碼不在討論的範圍內。以下為相同發票下一維條碼與二維條碼的解析程度。

開始實驗時，當相片傳進 ZXing，它所呈現出的辨識率趨近於 0。把相片傳進 Zbar 裡，所得到的辨識率也僅有 44.6%。而當放大之後，使用了手機上的 APP（QR Code 碼掃描器）來一張張掃描時，發現它有機會辨識出所有條碼。由此可知，ZBar 與 ZXing 在處理相片的 Resampling 方面還不夠完善，如果 pixel 不足，會發生 truncation error 的問題，而使得辨識率降低。因此，我們進而使用到了 Interpolation Filter 的工具。以下為 QR Code 經過拍攝後放大的 Interpolation Filter 與原始圖比較：

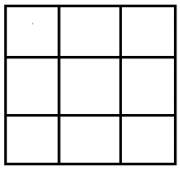
上圖為相片放大 Finder Patterns 後沒有 up sampling 與 Interpolation Filter 的原始圖。一張相片包含 50 多張發票，放大後的模糊程度，從 Finder Patterns 就可以觀察出圖形的粗糙，最終導致辨識率不良。

上圖為 Lanczos Filter 內插取樣的圖形，跟上面的原始圖比起來，經過 up sampling 後像素多出一倍，解析度明顯提升，對於解碼方面較容易辨識。

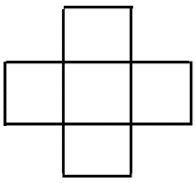
2.改進 qr_img_bit() 中取樣的演算法

一個 module 大約是由數個像素所組成，當系統跑到 qr_img_bit() 函式時，會開始進行取樣，只要能判斷 module 內所有像素的黑或白，就能決定出 module 的值。

在原本 qr_img_bit() 函式中每個 module 的取樣方式為，在該 module 的估計範圍內，取一個像素來當作是整個 module 的值 (黑或白)。此方法所得到的結果在解析度不高時，比較容易因為雜訊，使得黑點或白點皆容易變成灰點，而產生誤判。因此，底下提出不同的演算法，想辦法提升取樣的正確率。



九宮格的方式取樣。以中心與鄰近的八個像素做取樣，加值的總合若低於平均則將此 module 視為白色，若高於則視為黑色。以此方法來彌補隨機取樣正確率的不足。



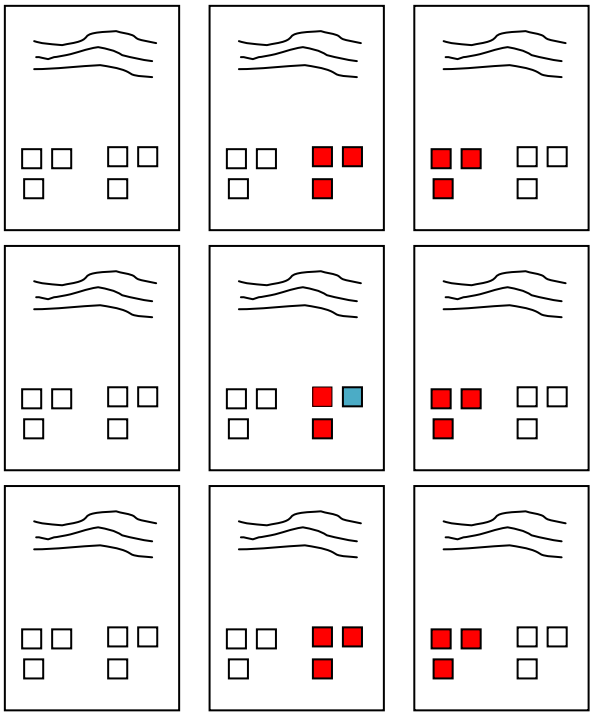
十字宮格取樣。以中心為準，分別取它上下左右的像素，若垂直的三個像素平均值較高 (黑色)，就拿垂直的像素來做取樣，否則拿水平的像素做取樣。以此方法來彌補隨機取樣正確率的不足。

3. 最佳化 qr_reader_match_centers()中排序的演算法

此函式的目的為，對所有的 Finder Patterns 透過暴力搜索(Exhaustive Search)進行組合比對，判斷是否為一個正確的 QR Code。在一個正確的 QR Code 中包含了 3 個 Finder Patterns，其總共需要比對的次數為C_3^n，時間複雜度為 O(N^3)。而放入本研究的相片裡(112 個 QR Code、336 個 Finder Patterns)，比對次數則是C_3^{336}，共超過 620 萬筆配對組合。此步驟因配對組合過於龐大，導致運算時間過長的問題。因此，底下提出不同的演算法，以解決其不足之處。

本研究將此演算法中的排序部分最佳化，將原本的隨機排列順序改成以距離重新排序，由最近的一圈 Finder Patterns 來依序比對(約取 20 個左右)。並使用了曼哈頓距離法，計算距離的方式為 |X+Y|的相對算法，取代了計算絕對距離所需花費的時間。最後比對次數不超過 20 * 20，與原先的C_3^n相比，節省了大量的時間。助於我們進一步的減少隨機比對組合的次數。

以下圖做為舉例：



其中有 9 張發票，每張發票有 2 個 QR Code，共計 6 個 Finder Patterns(小方塊)。若為隨機配對組合，則所有的 Finder Patterns 都會與中間藍色標記的 Finder Patterns 配對(C_3^n)。而本研究的方法為運用曼哈頓距離法，找出相對較近的 Finder Patterns(紅色)進行配對(20 * 20)，避免掉絕大多餘的組合，縮短了大量的時間。

八、實驗設計

（一）樣本來源：以 Nexus 6P 手機拍攝 4 組發票相片 ，每張照片有 56 張發票(112 張 QR Code)，照片解析度為 (3984 * 5312)

（二）驗證 up sampling 對於辨識率的影響

1. 分別以 14 種不同的 Filter 對樣本照片做 up sampling
2. 實測 up sampling 之後照片的辨識率
3. 比較其與未做 up sampling 之原始相片的辨識率差異

（三）驗證改進後的 qr_img_bit()演算法對辨識率的影響

1. 分別採用九宮格與十字宮格之演算法，對各種 Filter 進行過 up sampling 之相片進行辨識
2. 比較採用九宮格、十字宮格與原始演算法的辨識率差異

（四）驗證改進後的 qr_reader_match_centers()演算法對辨識率的影響

1. 以改進排序過後的演算法，經過 up sampling 與 qr_img_bit()演算法的情形下，對於辨識率的影響
2. 比較其與原始相片的辨識率差異
3. 比較其與原始相片的時間效率差異

（五）驗證在相片放大處理後，經過 up sampling、qr_img_bit()與 qr_reader_match_centers()演算法的情形下，對辨識率的影響

1. 分別採用 1.5 倍與 2 倍的相片放大，對各種 Filter 進行過 up Sampling、qr_img_bit()和 qr_reader_match_centers()演算法之相片進行辨識
2. 比較採用相片像素 1.5 倍、2 倍與原始相片(1 * 1)的辨識率差異

（六）驗證在相片進行模糊化後，經過 up sampling、qr_img_bit()與 qr_reader_match_centers()演算法的情形下，對於辨識率的影響

1. 分別採用 1 倍、2 倍與 3 倍的模糊程度，對各種 Filter 進行過 up sampling、qr_img_bit()和 qr_reader_match_centers()演算法之相片進行辨識
2. 比較採用模糊度 1 倍、2 倍、3 倍與原始相片的辨識率差異

（七）驗證在相片進行銳利化後，經過 up sampling、qr_img_bit()與 qr_reader_match_centers()演算法的情形下，對於辨識率的影響

1. 分別採用 1 倍、2 倍與 3 倍的銳利程度，對各種 Filter 進行過 up sampling、qr_img_bit()和 qr_reader_match_centers()演算法之相片進行辨識
2. 比較採用銳利度 1 倍、2 倍、3 倍與原始相片的辨識率差異

伍、研究結果

一、驗證 up sampling 對於辨識率的影響

實驗過程以下列四張相片(3984 * 5312)做為樣本，一張相片中包含了 56 發票，其中每張相片的 QR Code 數量有 112 個。


相片 1


相片 2

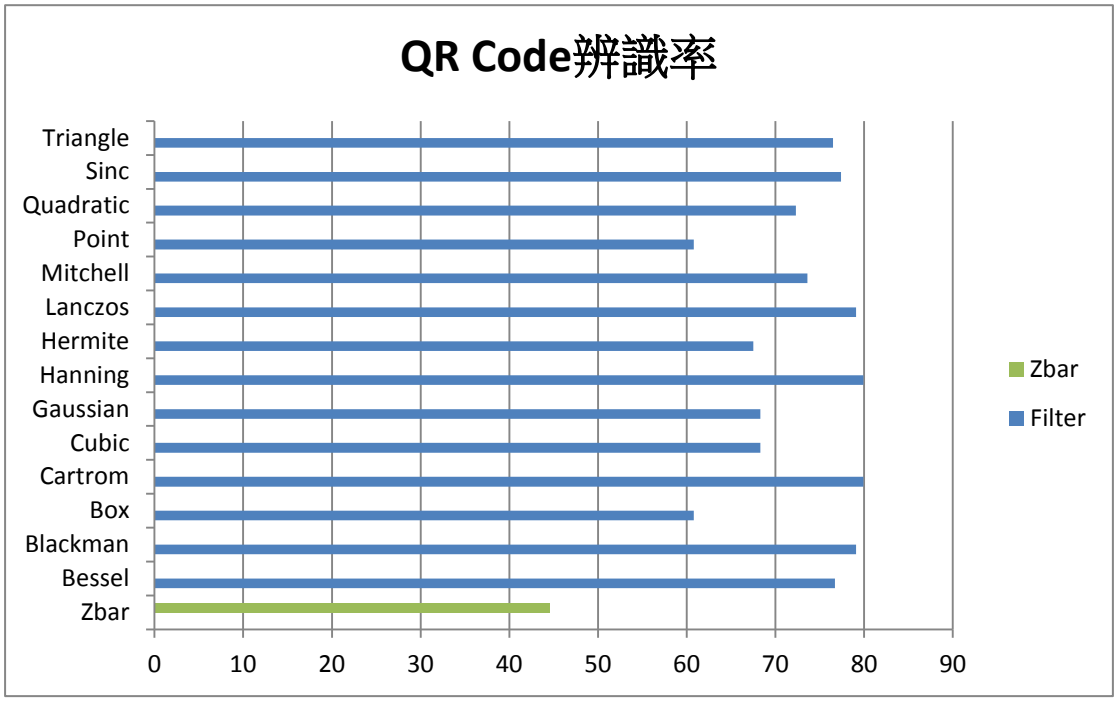

相片 3


相片 4

以下是我們利用各種 ImageMagicK 中內建的 Interpolation Filter 所做出的幾組實驗數據，數字代表其所正確辨識出的 QR Code 數量：

Filter \ 編號	相片 1	相片 2	相片 3	相片 4
Triangle	89	92	81	81
Sinc	96	80	85	86
Quadratic	82	82	80	80
Point	69	71	70	63
Mitchell	83	87	85	75
Lanczos	91	84	90	90
Hermite	83	74	74	72
Hanning	92	88	94	84
Gaussian	74	79	81	72
Cubic	79	73	82	72
Catrom	91	86	96	85
Box	69	71	70	63
Blackman	94	83	87	91
Bessel	84	92	87	81
No Filter	49	51	53	47

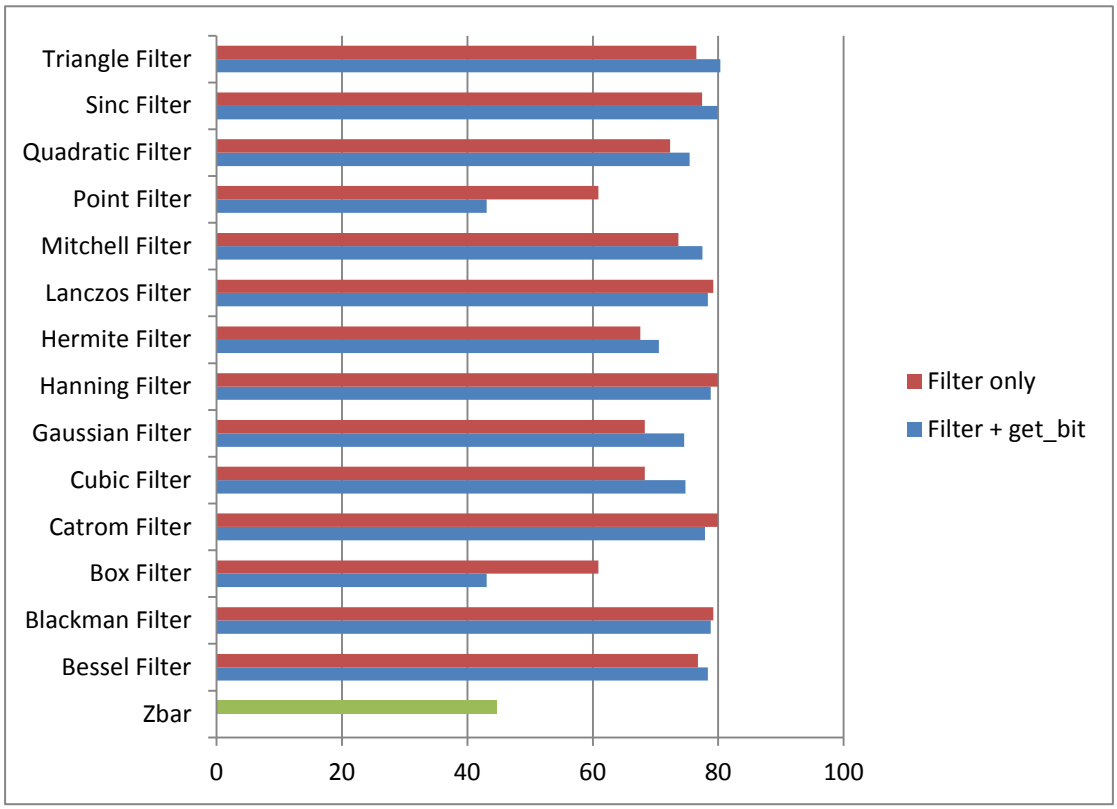
根據以上的幾組數據，將每一個的 Interpolation Filter 統整出以下圖表：



由此圖可見當我們加入 Interpolation Filter 後，QR Code 的辨識率大幅提升。拿 Lanczos 做舉例，在 No Filter 的情況下，Zbar 僅有 44.6%，當加入 Lanczos Filter 後辨識率達到了 79.1%，提升了 34.5%。其它的 Interpolation Filter 也有所改善。以上結論可以知道，增加 Interpolation Filter 的內插可以增強系統辨識，取代了原本 Zbar 與 ZXing 的辨識率。

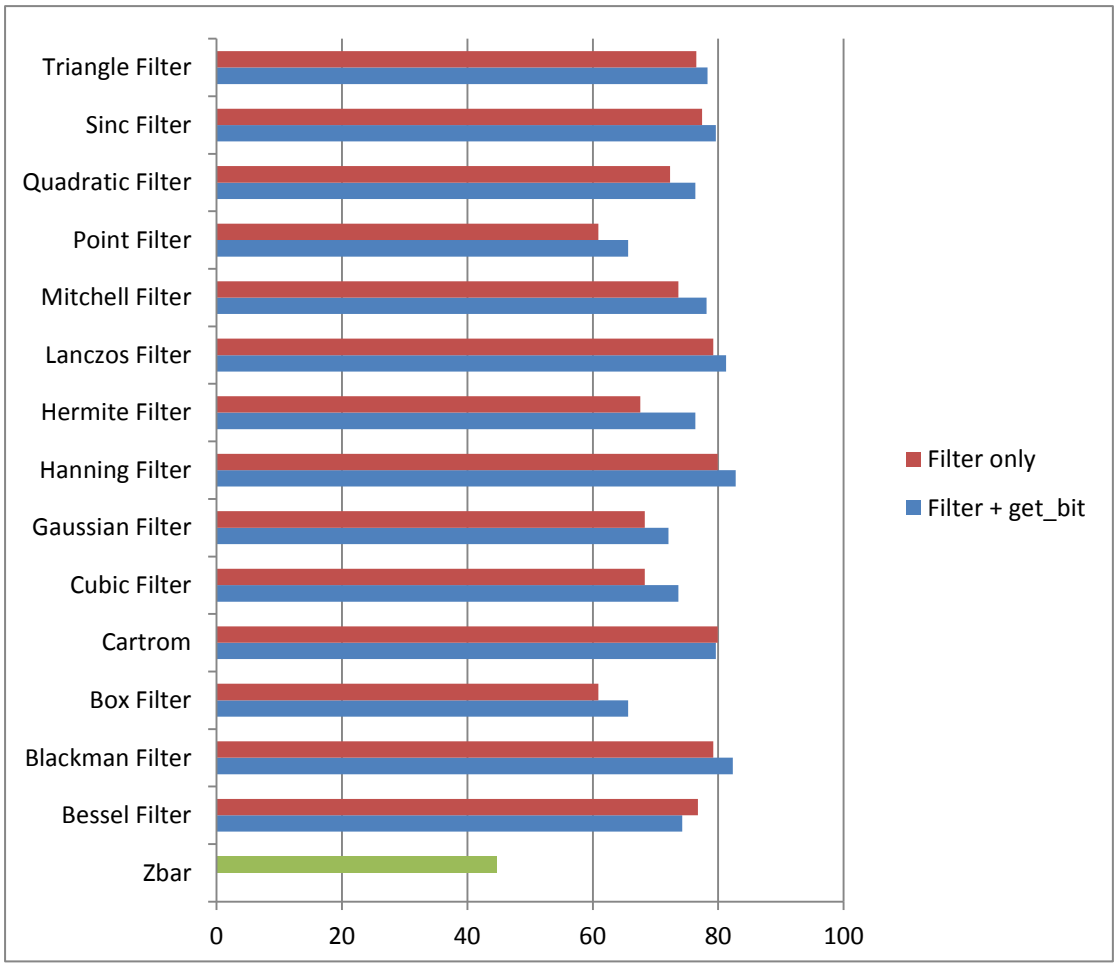
二、驗證改進後的 qr_img_bit()演算法對辨識率的影響

以下為隨機取樣與九宮格取樣法實驗後的比較數據 (紅色為隨機取樣，藍色為九宮格取樣)：



實驗結果發現，將隨機取樣的方法改為九宮格取樣並沒有太多明顯改善，反而部分 Interpolation Filter 的辨識率還有所下滑，並無法提升對 QR Code 的辨識率。

以下為隨機取樣與十字宮格取樣法實驗後的比較數據 (紅色為隨機取樣，藍色為十字宮格取樣)：



實驗結果發現，用十字宮格的取樣法平均能提升每個 Interpolation Filter 3.35% 的辨識率。如此可以消除一些隨機取樣所導致的誤判，進一步改善了對 QR Code 的辨識。

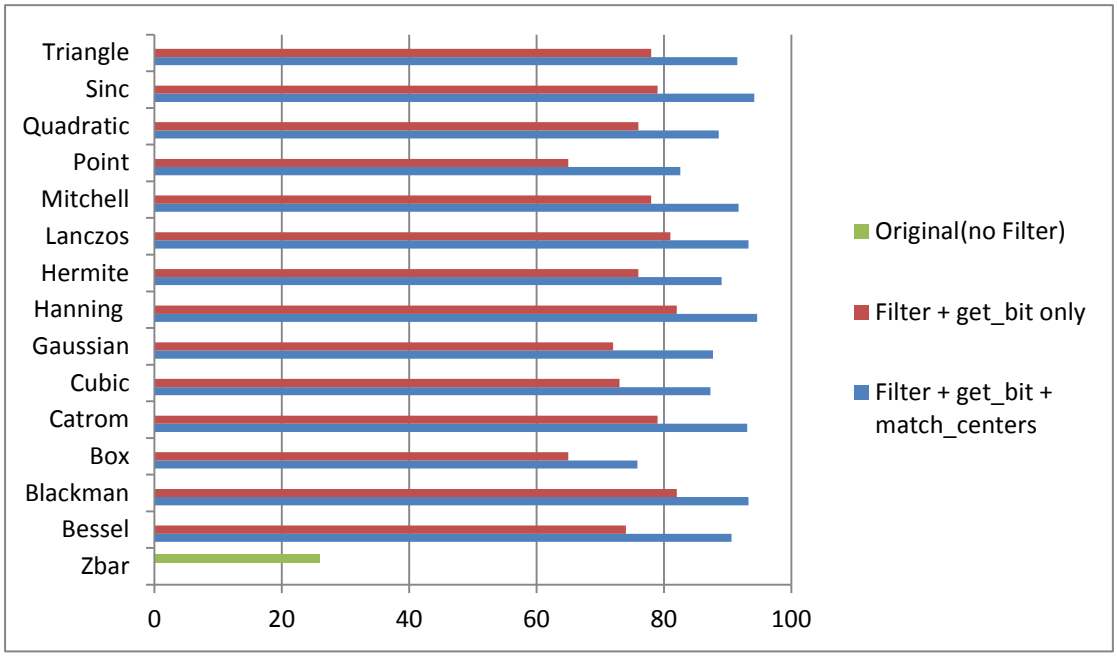
以下是我們利用各種 ImageMagicK 中內建的 Interpolation Filter 及十字宮格取樣法所做出的幾組實驗數據，數字代表其所辨識出的 QR Code 數量：

Filter \ 編號	相片 1	相片 2	相片 3	相片 4
Triangle	92	93	90	76
Sinc	84	91	97	85
Quadratic	89	82	86	85
Point	71	71	79	73
Mitchell	86	88	94	82
Lanczos	88	93	92	91
Hermite	89	85	86	82
Hanning	93	90	94	94
Gaussian	87	78	80	78
Cubic	88	81	88	73
Catrom	96	93	85	83
Box	71	71	79	73
Blackman	93	93	91	92
Bessel	88	85	87	73
No Filter	30	27	31	29

由以上兩個的實驗數據可以發現在沒有 Interpolation Filter 的情況下，若使用了九宮格取樣或十字宮格取樣，皆會使系統的辨識能力下滑，問題的原因出在當沒有 up sampling 內插運算、增加像素下，去使用了九宮格、十字宮格取樣，會導致過程中因像素不足而取樣到其他 module 的像素，影響到本身的數值(黑或白)。當最後解碼出的資訊若超出 Reed – Solomon 的容錯範圍，就會辨識不出資料的原樣。

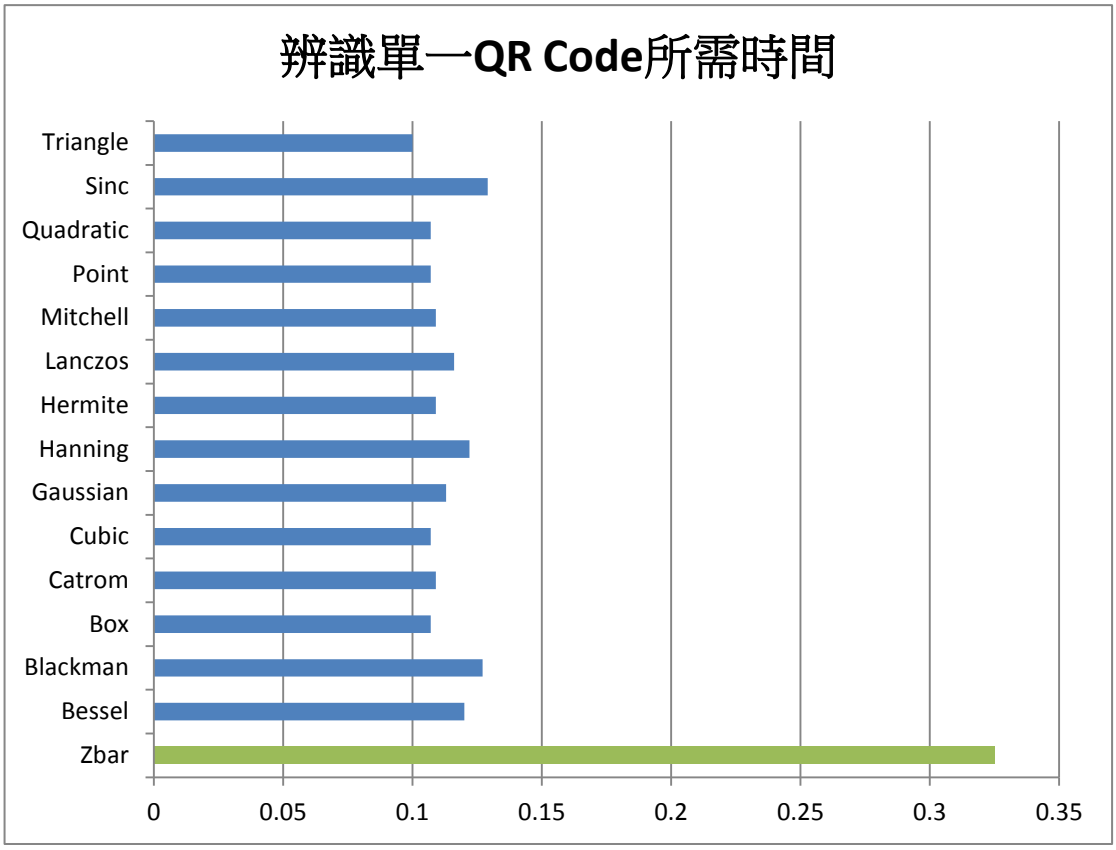
三、驗證改進後的 qr_reader_match_centers()演算法對辨識率的影響

隨後，探討將搜尋 Finder patterns 的演算法進行重新的排序與修改下，來影響實驗的辨識率與時間效率。以下的實驗數據為有無加上 qr_reader_match_centers()演算法的辨識率比較：



由以上的實驗數據可知，若將比對 Finder patterns 的排序方式改成由相對距離最近的來開始判斷，辨識率會大幅提升。拿 Lanczos 做為舉例，原本僅有的 81%辨識率，當加入 qr_reader_match_centers()演算法後，辨識率提升至 93.3%，跟原本相比，提升了 12%左右。而其他的 Filter 也有所改善。

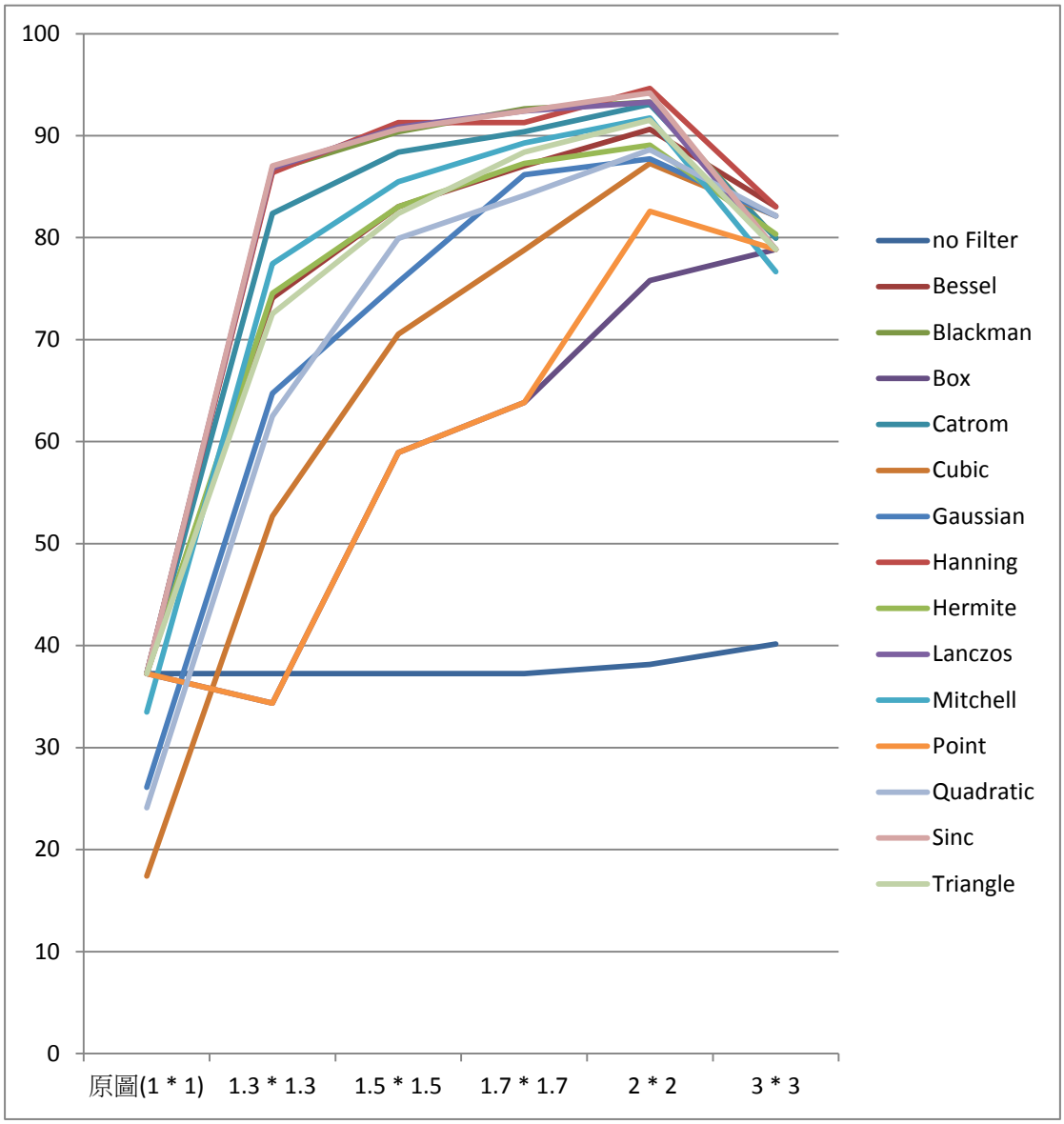
以下實驗數據是有無加上 qr_reader_match_centers()演算法的時間效率比較(以下為換算成平均每個 QR Code 的運算時間數據)：



實驗數據發現，原本的 Zbar 為 0.3 秒以上，當加入 qr_reader_match_centers() 演算法時，每個 QR Code 的運算時間縮短到 0.1 秒左右。在時間效率方面 Filter + get_bit + match_centers 方法明顯縮短許多。

四、驗證在相片放大處理後，經過 up sampling 、qr_img_bit()與 qr_reader_match_centers()演算法的情形下，對辨識率的影響

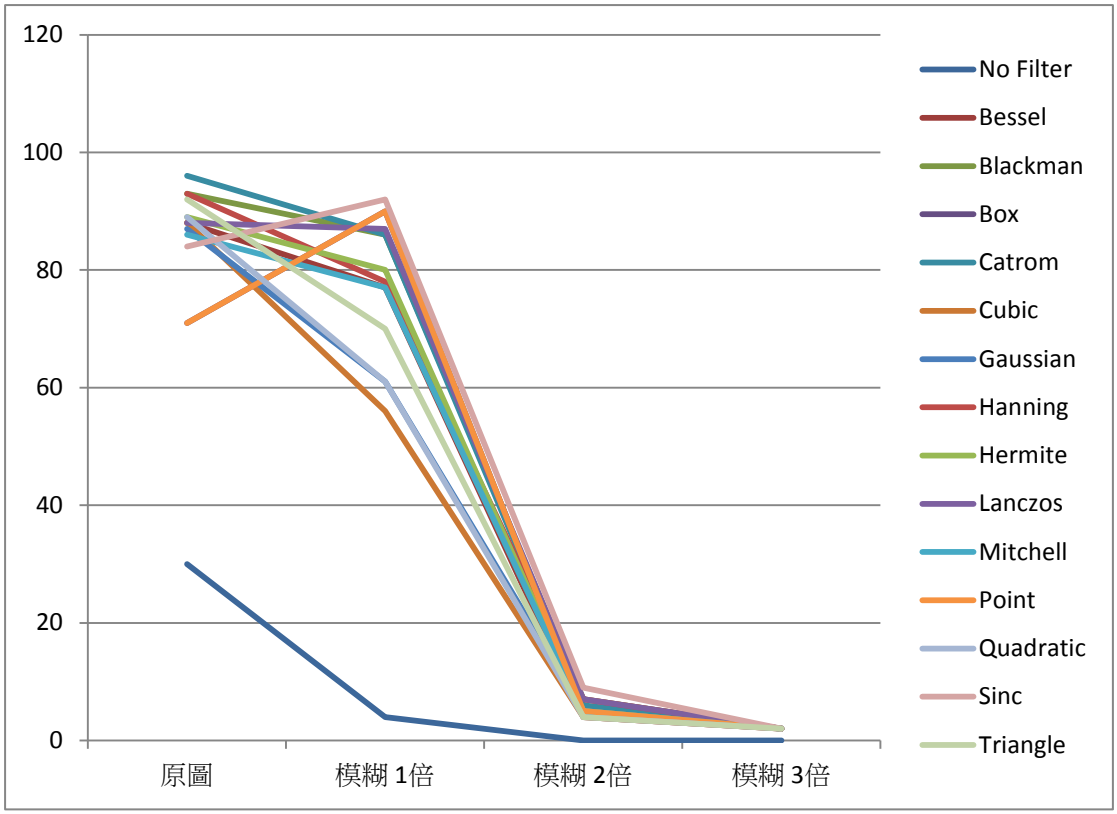
其中，也有探討將相片放大處理改變像素的數量，來影響實驗的辨識率。以下的實驗數據包含將長與寬乘以 1.3、1. 5、1.7、2 倍與 3 倍：



根據結果顯示，將相片放大 2 倍，能將辨識率提升到最高。若將相片放大到 3 倍，辨識率不但沒有明顯上升，且時間處理也比 2 倍時來的久。4 倍影像時間已耗時巨大而不列入考量。因此本研究決定以 2 倍的相片拿來做以上 Interpolation Filter、qr_img_bit()與 qr_reader_match_centers()中取樣演算法的實驗研究。

五、驗證在相片進行模糊化後，經過 up sampling、qr_img_bit()與 qr_reader_match_centers()演算法的情形下，對於辨識率的影響

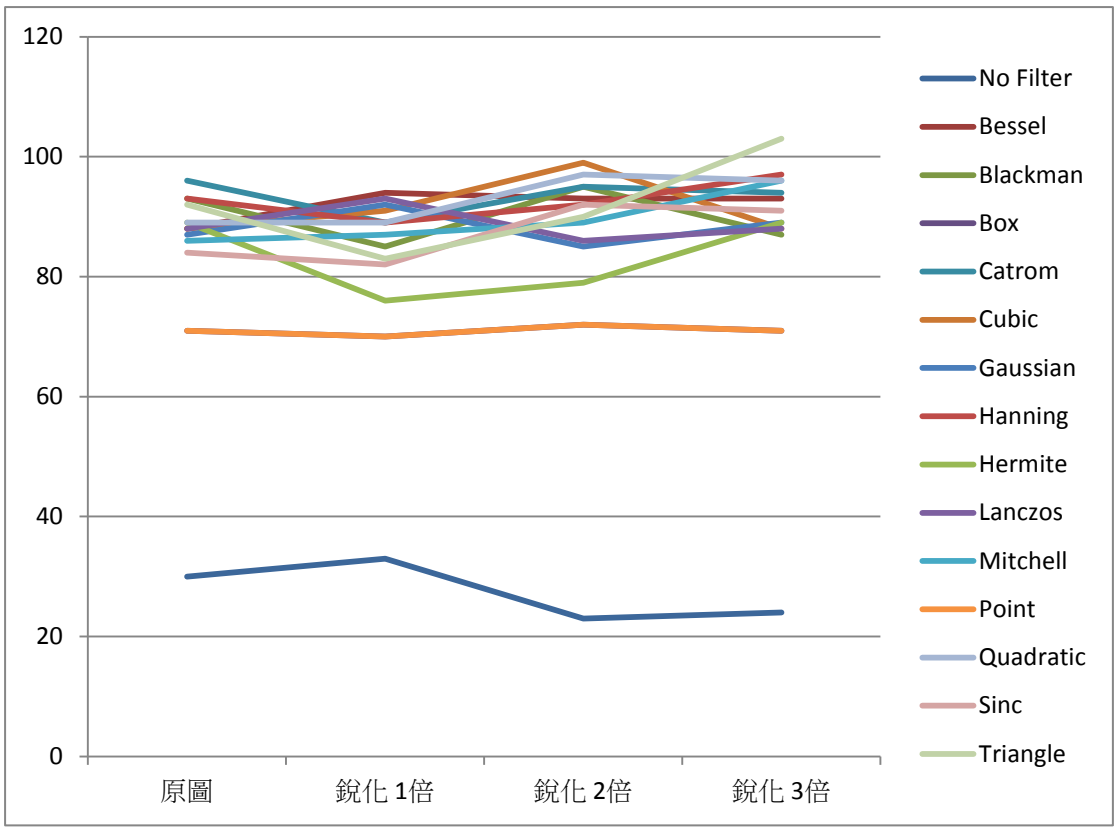
首先是調節相片的模糊度。以下是相片模糊後的實驗數據，模糊的程度包含 1 倍、2 倍與 3 倍：



實驗結果發現，當相片進行模糊化的時候辨識率會大幅下降，無法有效的提升 QR Code 辨識率。

六、驗證在相片進行銳利化後，經過 up sampling、qr_img_bit()與 qr_reader_match_centers()演算法的情形下，對於辨識率的影響

以下是調節相片銳化度的實驗。將相片銳化程度分為 1 倍、2 倍與 3 倍，並得到以下的實驗數據：



實驗結果發現，當相片進行銳化的時候辨識率沒有明顯的上升與下滑，和模糊的實驗一樣，無法有效的提升 QR Code 辨識率。

陸、結論

現今社會上，二維條碼是隨處可見，已經取代了一維條碼成為了主流，舉凡電子發票，電子支付，以及廣告促銷等的廣泛的應用。目前的條碼掃描 APP 都受限於僅能掃描單一 QR Code，對於一些需用掃描大量發票的使用者而言沒有實用性，能力、效率都有限。

因此，本研究以改善多個 QR Code 的辨識率為核心，當一張相片包含 112 個 QR Code 時，目前國外辨識率最高的 Zbar library，只能達到 44.6%。透過本研究提出之 up sampling 與 Interpolation filtering、module sampling、qr_reader_match_centers() 提升取樣準確率進行系統改善，辨識率可大幅上升至 93.3%，而 QR Code 平均運算時間也縮短為原本的 36%。未來希望能將成果回饋到 Zbar open source library，提供給國內外 APP 開發者更好的條碼辨識系統，讓使用者能高效率的完成掃描與解碼。

柒、參考資料及其他

一、參考資料連結

[1] 圖 1.Kun-Mao Chao & Kun Chen。淺談 QR Code。檢自：
https://www.slideshare.net/kchen/qr-code-61519555(Mar. 29, 2016)

[2] 中央警察大學資訊密碼暨建構實驗室（ICCL）。網管人。檢自：
https://www.netadmin.com.tw/article_content.aspx?sn=1304030005(Apr. 4, 2013)

[3] Wikipedia。Reed–Solomon codes for coders。檢自：
https://en.wikiversity.org/wiki/Reed%E2%80%93Solomon_codes_for_coders

[4] 圖 2.QR code.com。Error correction feature。檢自：
http://www.qrcode.com/en/about/error_correction.html

[5] 圖 3.Kun-Mao Chao & Kun Chen。淺談 QR Code。檢自：
https://www.slideshare.net/kchen/qr-code-61519555(Mar. 29, 2016)

[6] 圖 4.Keith Noah Snavely。Cornell Computer Science。檢自：
http://www.cs.cornell.edu/courses/cs6670/2011sp/lectures/lec03_resample.ppt

[7] 圖 5.Wikipedia。Lanczos resampling – Wikipedia。檢自：
https://en.wikipedia.org/wiki/Lanczos_resampling

[8] Wikipedia。Taxicab geometry。檢自：
https://en.wikipedia.org/wiki/Taxicab_geometry

二、條碼測試影像檔

相片 1


TAIPEI 101
電子發票證明聯
107年07-08月
FP-45712715
COSTCO
WHOLESALE
電子發票證明聯
107年7-8月
FH-37718059
一芳
電子發票證明聯
107年07-08月
FZ29713692
三和麵食店
電子發票證明聯
107年09-10月
HQ-83578286
McDonald's
電子發票證明聯
107年09-10月
HJ-15587477
McDonald's
電子發票證明聯
107年09-10月
HJ-15620027
電子發票證明聯
107年09-10月
HU-95916895

相片 2

相片 3

相片 4


FamilyMart
TAIPEI 101
107年07-08月
FP-45712715
McDonald's
McDonald's
7-ELEVEN

【評語】052503

此作品研究如何使用一些影像處理的技術來增加在同一張相片中正確辨認很多張 QR code 圖的正確率。這些處理技術包含一些 up-sampling 和 resampling 還有 Interpolation Filtering 等技術。作者設計和進行一些實驗來量測這些處理技術對提升辨識正確率的改進幅度，實驗設計豐富且實驗結果探討完整，實驗結果顯示使用作者所提出的方法可以大幅提升在同一張相片中正確辨認很多張 QR code 圖的正確率。

建議考慮使用 segmentation 技術先對同一張相片中的多張 QR code 進行切割，之後再對每一個切割出來的 QR code 進行所提出的處理方法，這樣可以大幅節省所需要的處理時間。

作品海報

摘要

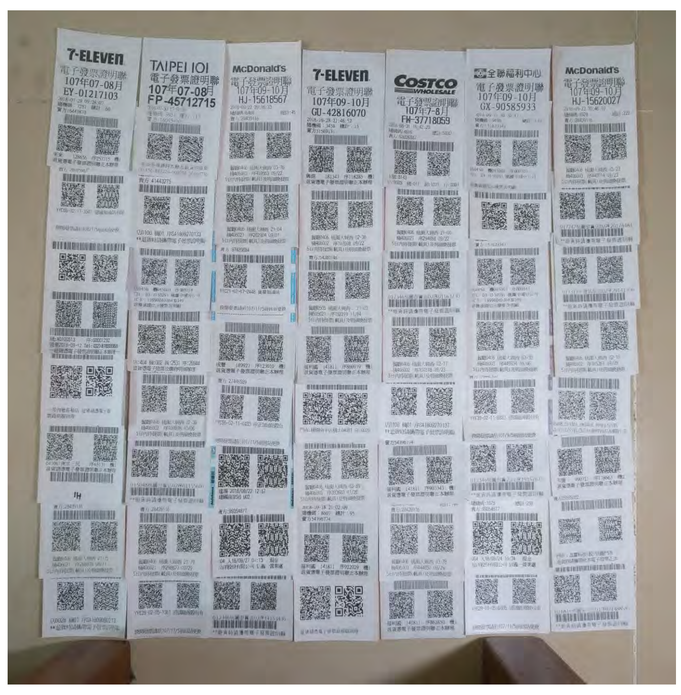
本研究針對開源QR Code辨識函式庫目前辨識率最高的Zbar library作改進，發現使用up sampling搭配多種不同Interpolation Filtering，可以有效提升多個QR Code的辨識率。如果再加上本研究提出的module sampling與match centers方法，這樣一張相片 中包含112個QR code的辨識率從原本44.6%可以提升至93.3%，而平均QR Code辨識時間也大幅縮短為原本的36%。

在近代的統一發票中，因為二維條碼資料容量大、可附加logo標商效果且具有容錯機制，所以使用方面逐漸取代了一維條碼。

再者，雖然發票中也有一維條碼可以掃描，但因為一維條碼解析度太低，再加上沒有容錯機制，所以在掃描的過程中條碼辨識不出來。因此一維條碼不在討論的範圍內。

動機

對於一些需用掃描大量條碼的使用者來說，如果一張張掃描的話，將耗費龐大的時間。 因此，本研究設計與實作出能掃描
大量QR Code的有效方法。
以慈善基金會為例，近年來，每年的國人捐獻發票量都是以千萬張為單位，要在短時間內利用人力或一般手機的應用程式完成比對是極耗時間的。

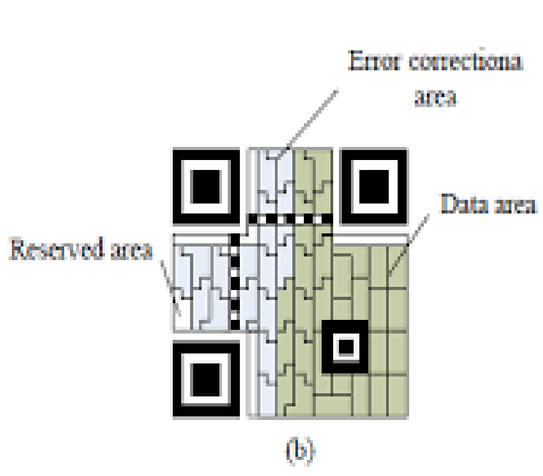


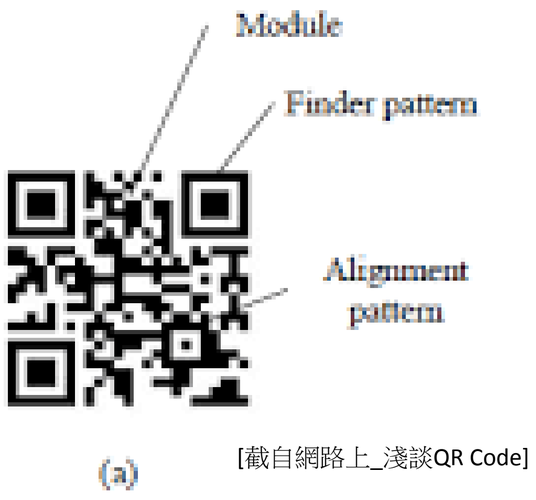
QR Code圖碼

QR Code圖碼成四方形，獲得IOS國際標準採用。其分別由以下部分所成：

（一）Finder patterns：由此可快速找到QR Code的位置。
（二）Alignment Patterns：可以提高QR Code的識別率。
（三）module：QR Code中最小的基本單位，由黑白碼元所組成（黑色表示1; 白色表示0），由此來組成真正的數據部分。
（四）Error Correction：進行一些錯誤的修正。

圖 [截自網路上_]電子發票證明聯二維條碼規格





[截自網路上_淺談QR Code]

程式流程圖

研究過程

利用ImageMagicK中內建的Interpolation Filters所做出的幾組實驗數據
1. 本實驗分別以14種不同的Filter對樣本照片 做up sampling
2. 比較其與未做up sampling之原始相片 的辨識率差異

紅色曲線

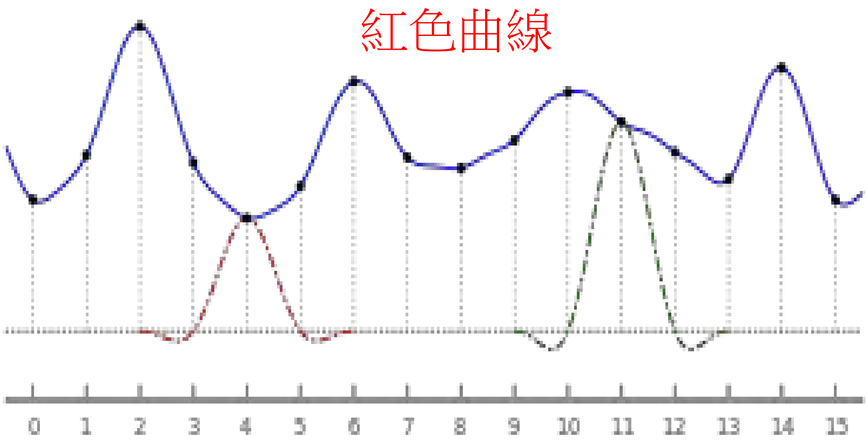
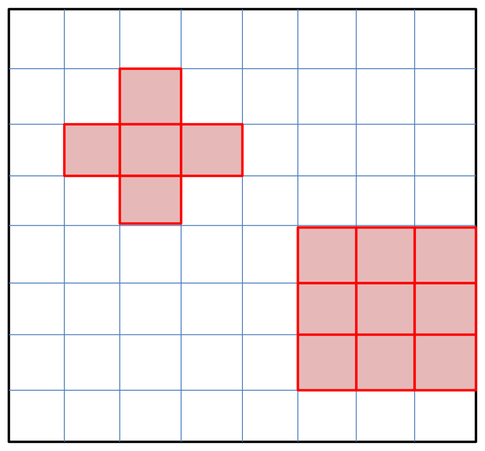


圖 [截自網路上_維基百科]

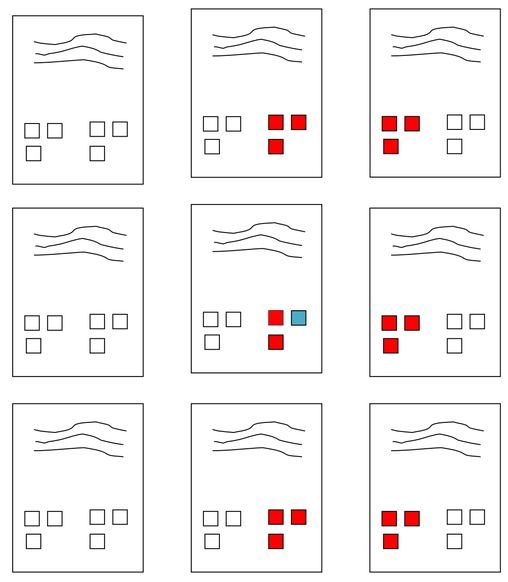
拿Lanczos做舉例，在No Filter的情況下，Zbar僅有44.6%，當加入Lanczos Filter後辨識率達到79.1%，提升了34.5%。其它的Interpolation Filter也有所改善

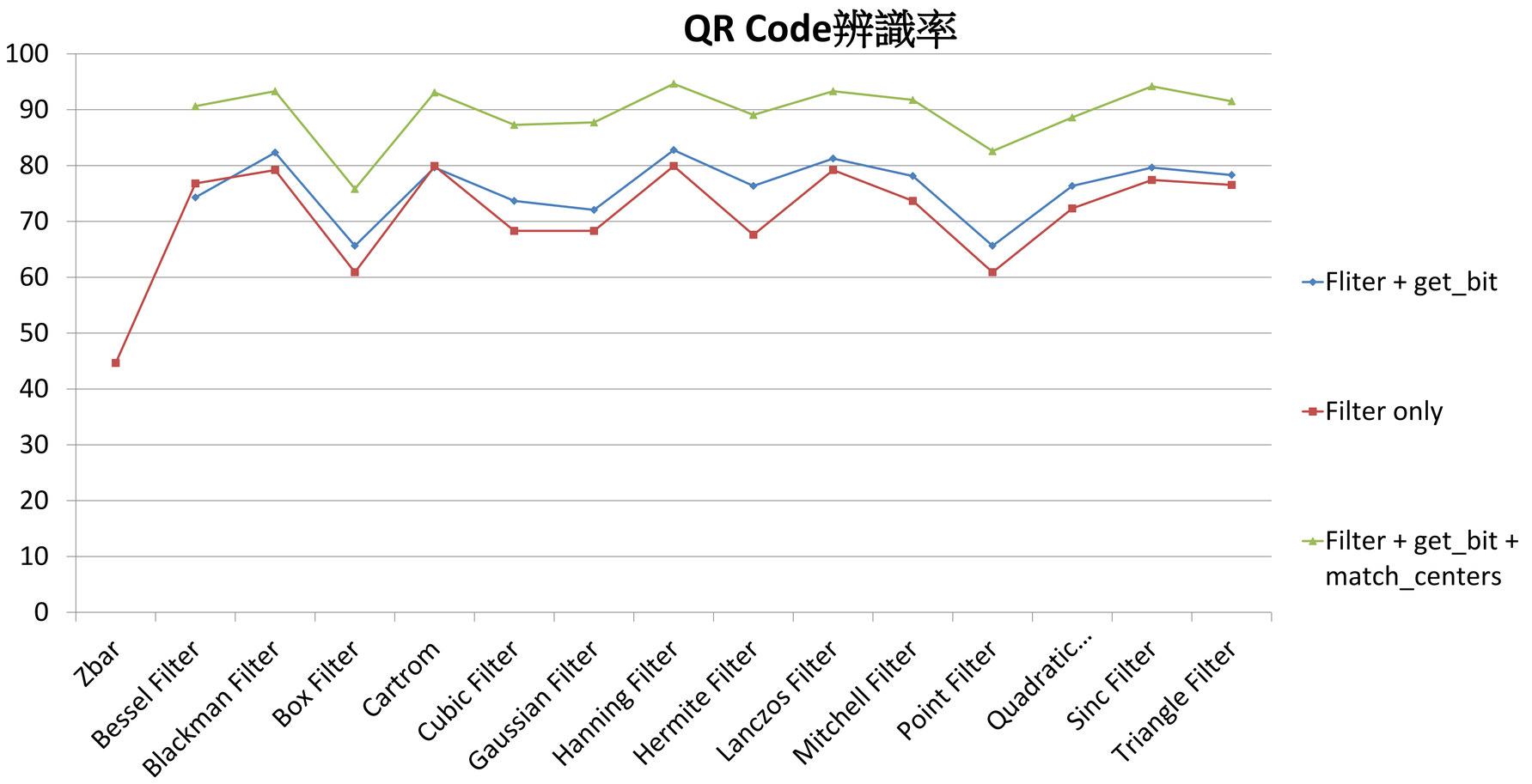


藍色曲線

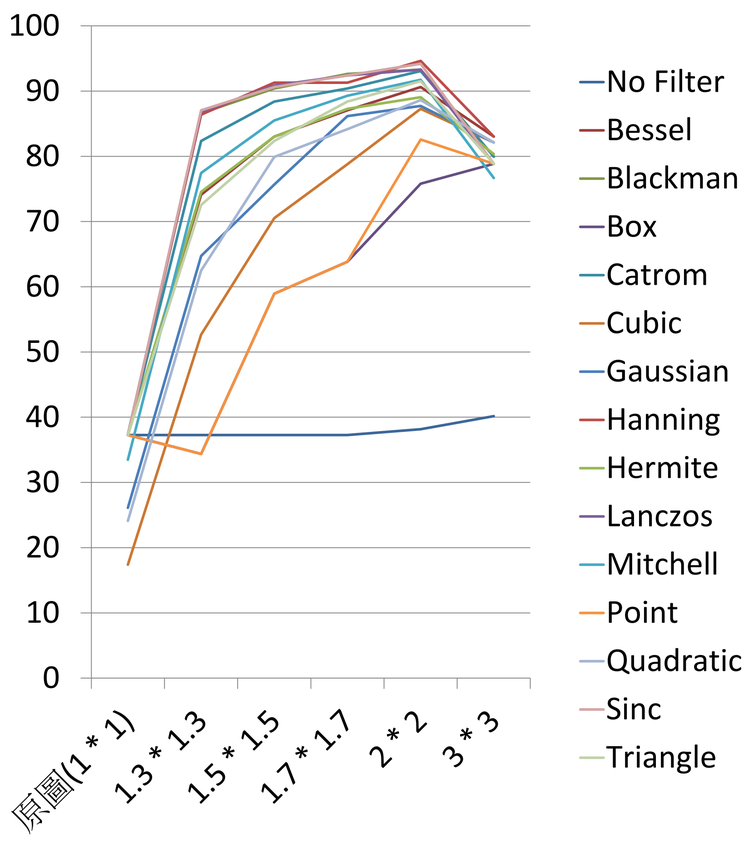
1. 採用十字宮格之演算法，對各種Filter進行過 up sampling 之相片 進行辨識
2. 用十字宮格的取樣法平均能提升每個Interpolation Filter 3.35%的辨識率。

綠色曲線



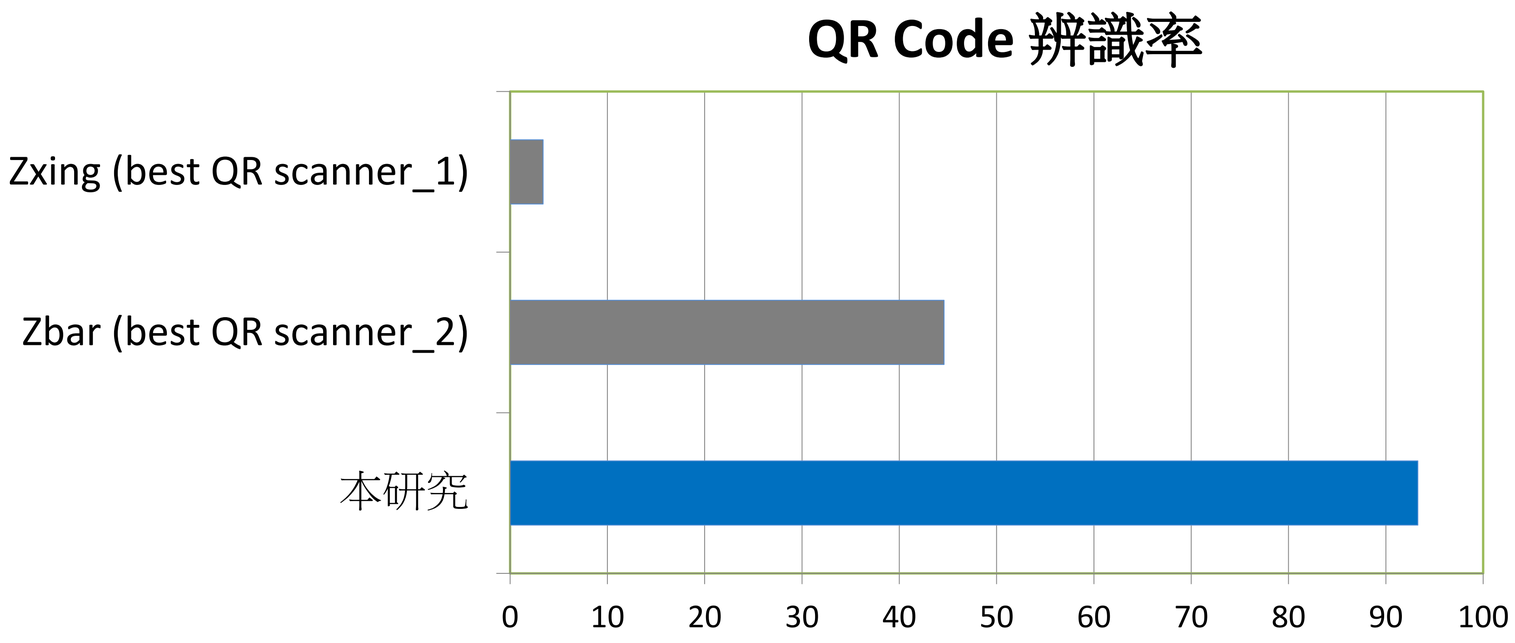


實驗數據發現，原本Zbar判斷單一QR Cdoe平均時間為0.3秒以上，當加入qr_reader_match_centers()演算法，單一QR Code的運算時間縮短到0.1秒左右。在時間效率方面Filter + get_bit + match_centers方法明顯縮短許多。

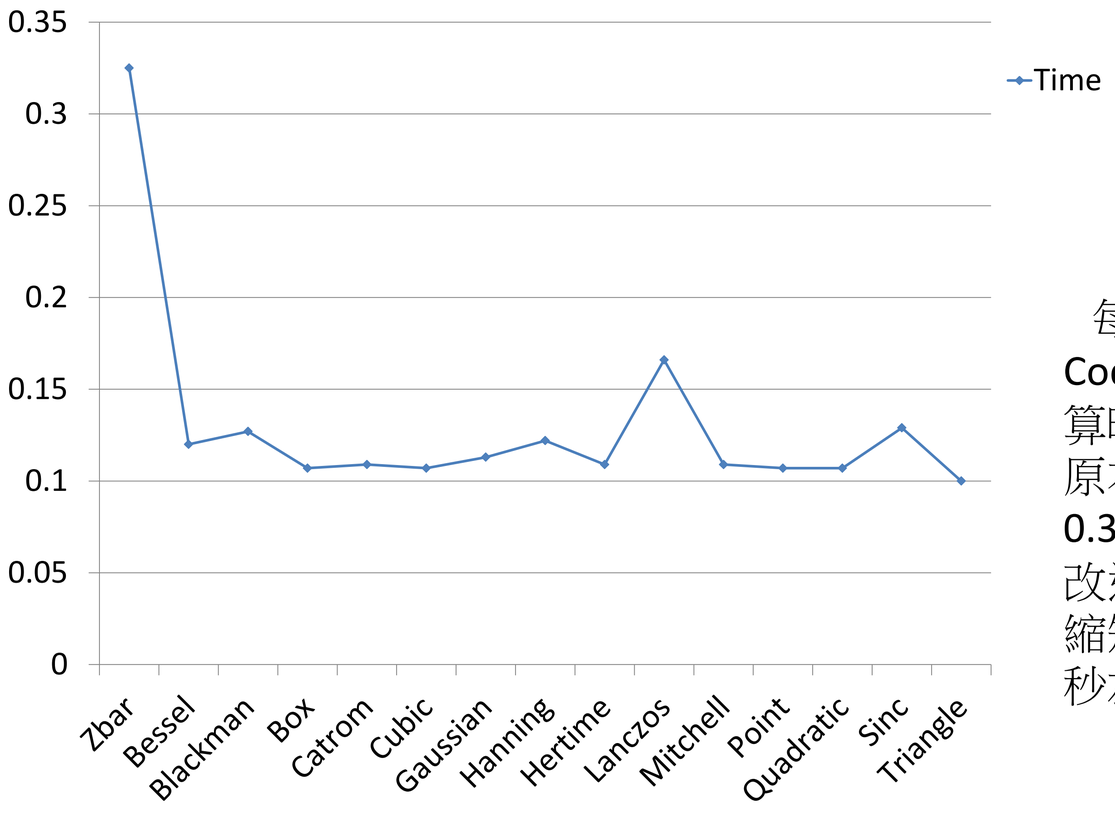


根據結果顯示，將相片 放大2倍，能將辨識率提升到最高。若將相片 放大到3倍，辨識率不但沒有明顯上升，且時間處理也比2倍時來的久。

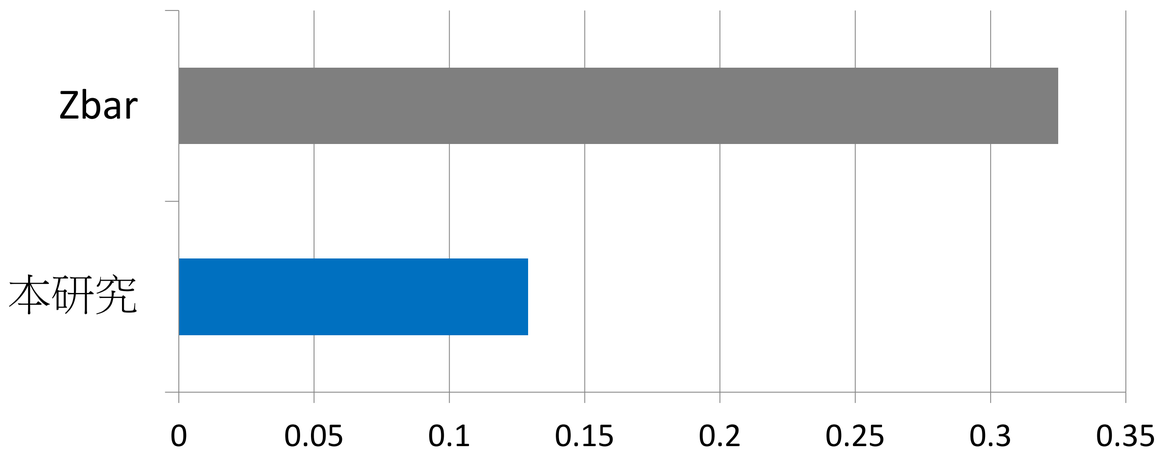
QR Code 辨識率從 44.6% 提升至 93.3%



辨識時間從 0.33 秒縮短至 0.13 秒



每個QR Code的運算時間，原本Zbar為0.3秒以上。改進後，縮短到0.1秒左右。



Reed–Solomon 介紹

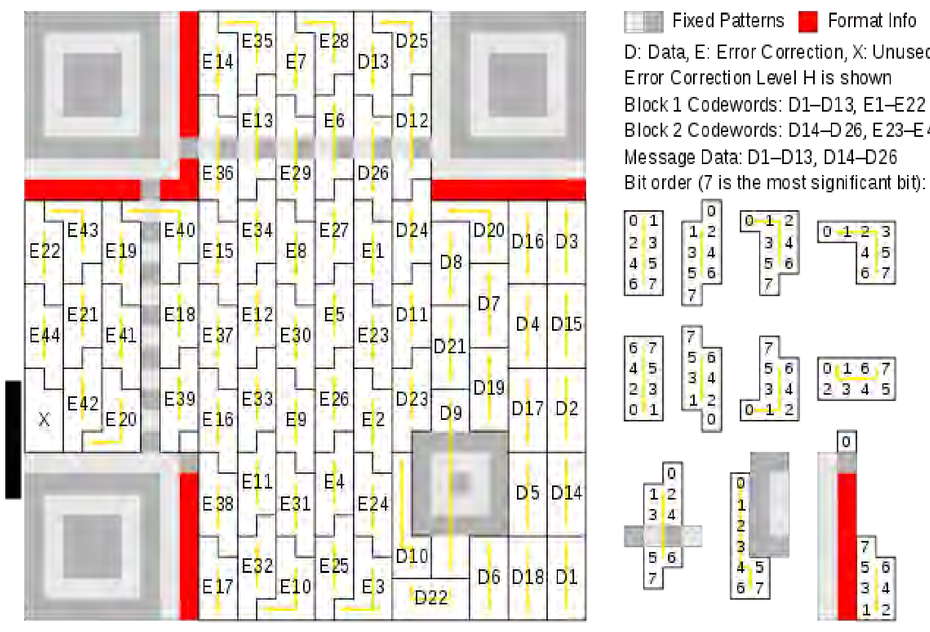


圖 [截自網路上_ **Reed-Solomon Error Correction**

Reed–Solomon 是個非常重要的錯誤更正碼，廣泛的運用在各個通訊上。四種容錯等級，分別為L、M、Q、H，不同的等級，可接受的損毀率也更高。適用的等級也會隨著資料含量有所調整。

圖為 QR Code 資料放置結構，其中D為條碼所包含的原始資訊，E則為Reed–Solomon 中所需用到的備份字典。

QR Code Error Correction Capability*	
Level L	Approx 7%
Level M	Approx 15%
Level Q	Approx 25%
Level H	Approx 30%

圖 [截自網路上_ Error Correction Feature

THIS
THAT
CORN

THISABCD
THATBCDE
CORNCDEF

如果掃描出來的資料為th**時，備份字典中會有重複字母的問題，而無法恢復完整的訊息。解決此問題的最簡單方式，就把每個單字後面加長，讓每個單字都是獨特的。

在單字後面都進行增長，使this與that可以運用多餘字元來恢復原始數據。

圖 [截自網路上_]電子發票證明聯二維條碼規格

參考財政部規定，左方二維條碼應使用QR Code V6(41x41)(含)以上版本，並採用Level L(容錯率7%)以上之防錯標準。

Truncation error

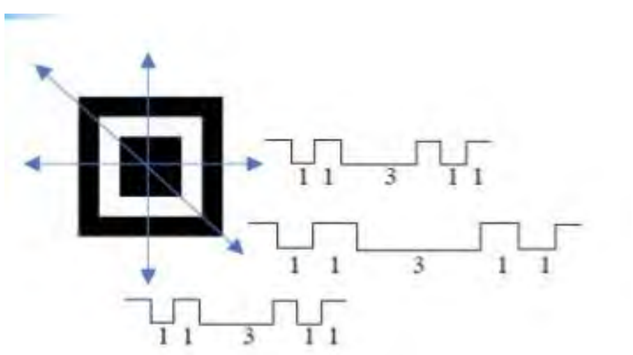
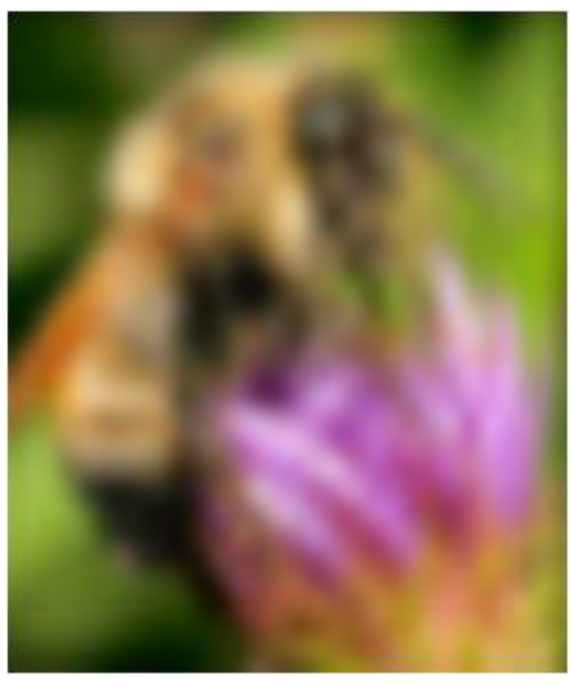


圖 [截自網路上_淺談QR Code]

Truncation error為Finder Patterns的格式判斷。一般的格式比例為1：1：3：1：1，當判斷的比電子發票證明聯二維條碼規格例錯誤或像素不足導致辨識不清時，就會出現Truncation error。

不同內插法的呈現



如圖，同樣都是將10倍小的原圖 up sampling。左邊為 Nearest Filter 所處理過的圖案，與右邊的 Sinc Filter 相比，僅用肉眼就能辨識出圖形的優劣。

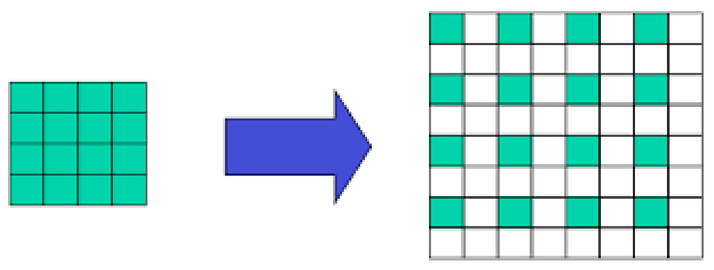
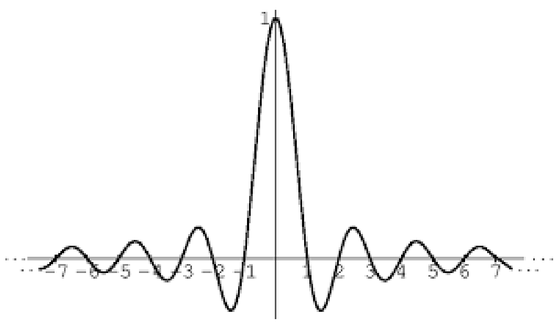


圖 [截自網路上_Upsampling and Interpolation]

[截自網路上_Cornell Computer Science]

Sinc 函數

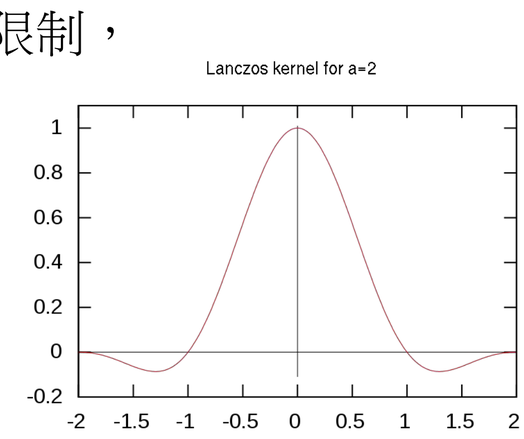


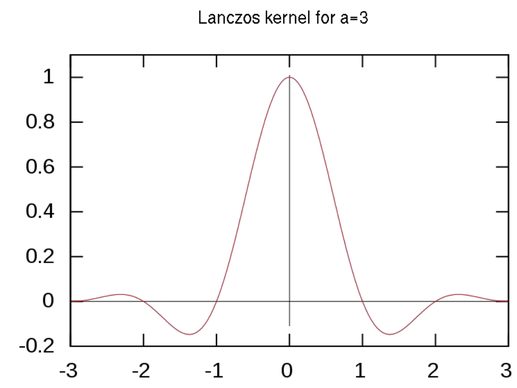
[截自網路上_維基百科]

Lanczos 普遍被認為是簡單filters裡的 “best compromise”，而 Lanczos 中的運算即為Sinc，唯一的差別在 Lanczos 有範圍限制，Sinc 則是無窮範圍。

對於所有 x ≠ 0，
$$Sinc(x) = \frac{\sin(x)}{x}$$

當 x = 0，
$$Sinc(0) := \lim_{x \to 0} \frac{\sin(ax)}{ax} = 1$$





[截自網路上_維基百科]

結論

目前辨識率最高的Zbar library，只能達到44.6%。透過本研究提出之up sampling與Interpolation filtering、module sampling以及match centers提升取樣準確率進行系統改善，辨識率可上升至93.3%。

透過本研究排序最佳化演算法，平均單一QR Code辨識時間由原本0.33秒大幅縮短為0.13秒。

本研究有助於一次掃描處理大量QR Code資料，預計回饋到 global open source 社群，使所有 Linux, Windows, Mac OSX, Android phone, iPhone, 及其他嵌入式系統等的 App 開法者，都得以將辨識率提升後的成果呈現給所有使用者。