逢　甲　大　學

資訊工程學系專題報告

醫學影像瀏覽系統

學　　生：劉奕廷（四乙）
顏嘉俊（四乙）
陳志玄（四乙）
黃耀田（四乙）

指導教授：黃秋煌　老師

中華民國九十二年十一月

目錄

圖表目錄

第一章　序論

一-1　實作動機

對於歷史文物書籍的保存不易等問題，可以透過影像圖片的方式，來做備份管理，而影像利用數位化的方式，可以加強影像的管理和保存性，醫學影像亦然如此，例如 X 光片、超音波掃瞄等影像。因此可實作一套系統，經由網路提供影像瀏覽及簡易的影像處理功能，達到線上臨床診斷的需要。

一-2　實作目標

在 web-based 的前提下，採用 server-client 的模式，透過 Java 程式語言跨平台的特性，利用安裝 Java 虛擬機器的 browser，不需事先安裝應用軟體，即可達到方便地進行影像瀏覽的目的，進而克服醫療影像診斷方面受限於空間及時間方面的因素。

第二章　系統分析與設計

二-1　工具的選擇

要達到讓使用者能遠端瀏覽自己所需要的圖，那就要達到遠端跟自己所需要這兩點；能選擇自己需要的圖，最簡便的方法是透過資料庫的幫助。為了此二主要目標所選擇的工具有：

網路⇒Apche、PHP、Java（Applet）。

資料庫⇒MySQL、PHP、Java。

以下列出選擇各工具的目的。

二-1.1　Apache 簡介

Apache 是非常有名的網路伺幅器軟體，由於他是免費的，其性能又是超強，所以是全世界最多人用的 Web Server。

二-1.2　選擇 Apache 原因

容易取得、免費、教學資源豐富、容易安裝使用。

二-1.3　簡介

PHP 是一種伺幅器端、跨平台, HTML 內嵌式描述語言。

二-1.4　選擇 PHP 原因

容易取得、免費、教學資源豐富、容易安裝使用、可讓網頁跟資料庫作連結、效率高。

二-1.5　Java 簡介

Java 是一種可攜式的物件導向語言，可攜式就是跨平台，且提供豐富的 API 元件讓寫程式的效率大大提升，是目前非常風行的程式語言。

二-1.6　選擇 Java 的原因

跨平台、API 元件豐富、可聯結資料庫（JDBC）、其 Applet 讓我們製作出所需的網頁功能，且可製作網路程式、Server 端的壓圖程式也是由 Java 撰寫。

二-1.7 MySQL 簡介

MySQL 是一個快速、多執行緒（multithread）、多使用者且功能強大的關聯式資料庫管理系統（relational database management system, RDBMS），可以與 C、C++、Java、Perl、PHP 等語言很容易的連結，可以運行於多種平台上，例如：Solaris、RedHat、Linux、FreeBSD、OS/2、Windows ..... 等等。

MySQL 在一般情況下需付費，不過，大體上來說，個人及非營利單位使用它是免費的。

二-1.8 選擇 MySQL 的原因

支援 SQL 語法、可與 PHP 跟 Java 連結、以目前的使用目的免費、教學資源豐富。

二-2　資料庫

總共分 4 個 table

1. 醫生

醫生 ID	密碼	科別	名字

主要用在登入子系統，科別則區分能看哪些圖

2.病人

病人 ID	名字	電話

要有地方解釋當查詢名子有 2 個以上同名的用電話作分別

3.圖片

病人 ID	日期	部位	科別	序號	備註

4.拍照紀錄

病人 ID	日期	科別

二-3　流程圖

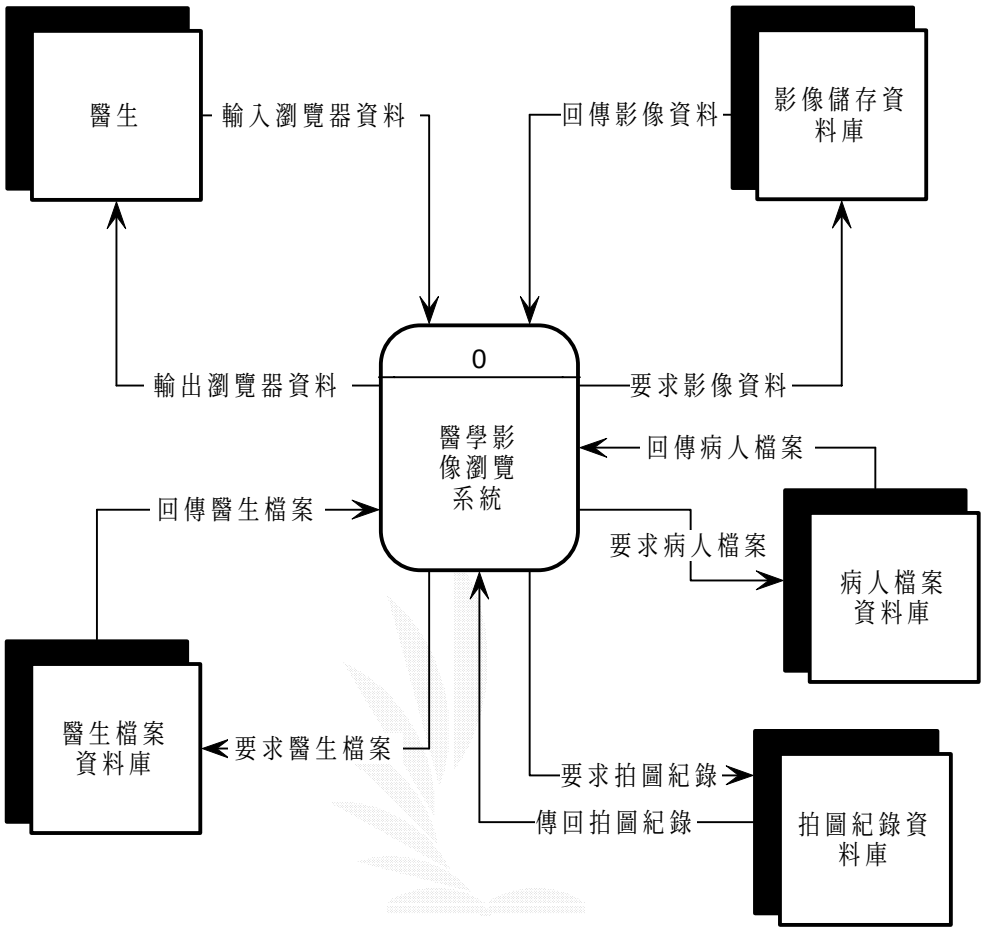


圖　二-3-1 Context Diagram

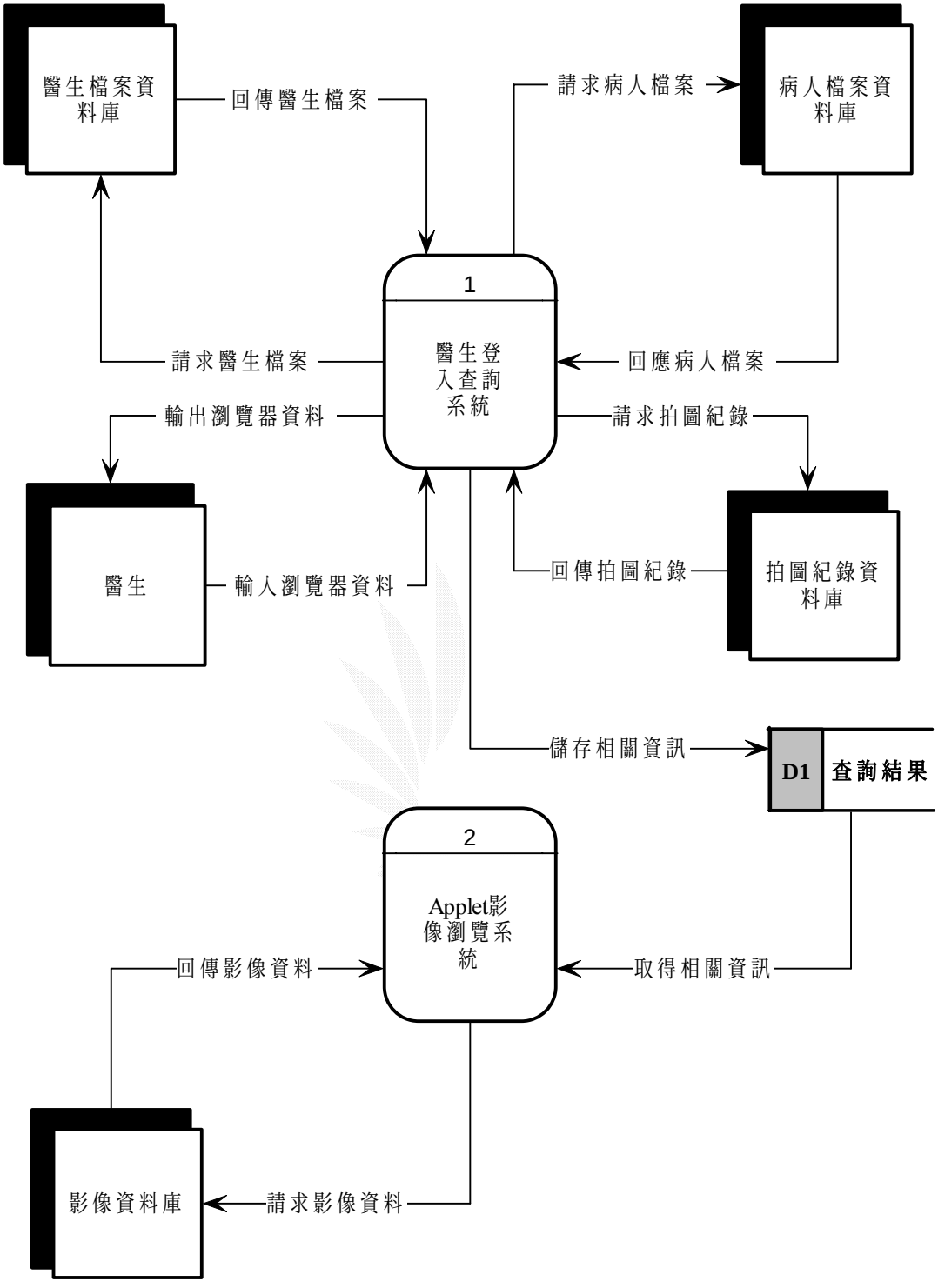


圖 二-3-2 Diagram 0

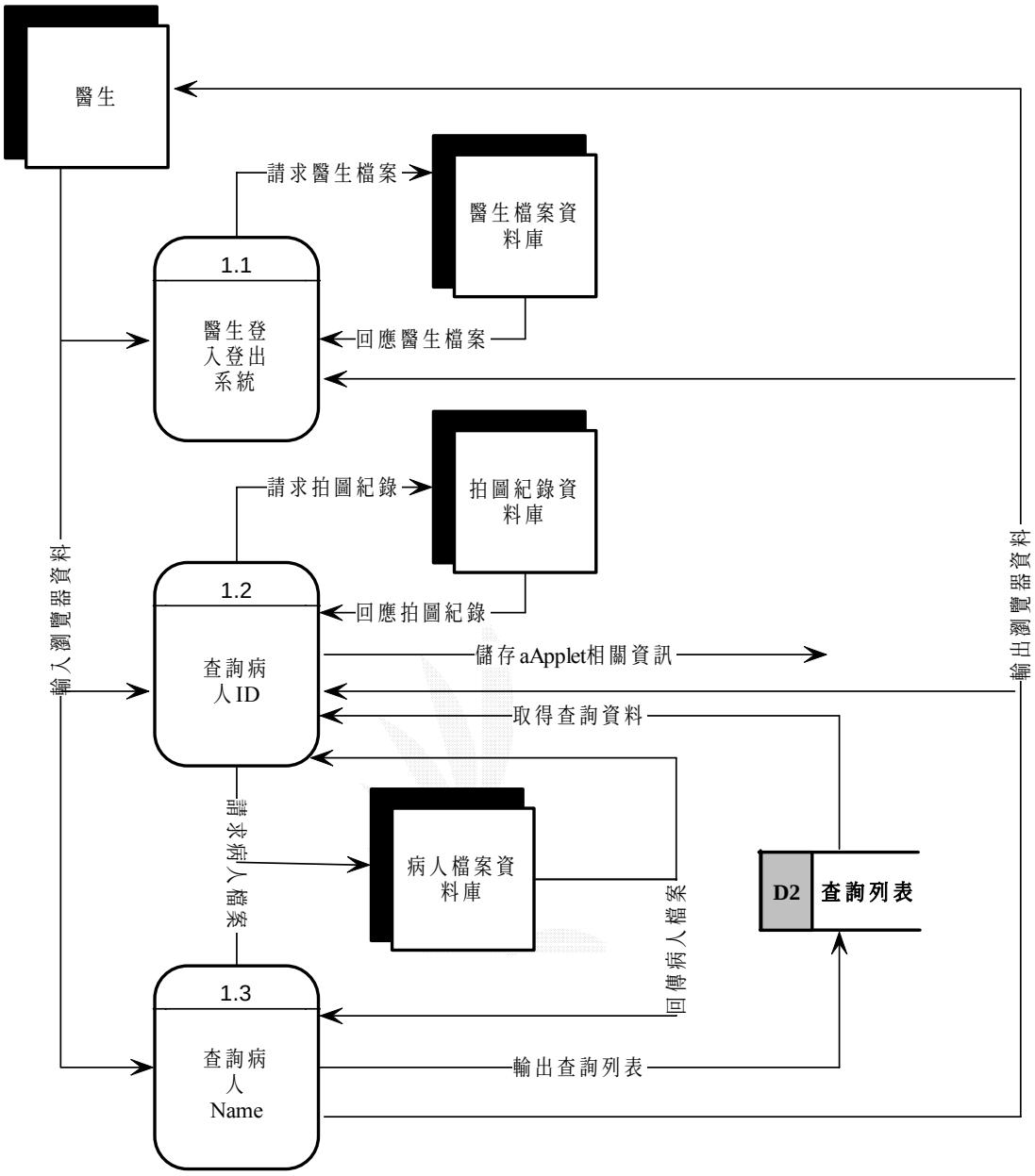


圖 二-3-3 DIAGRAM 1

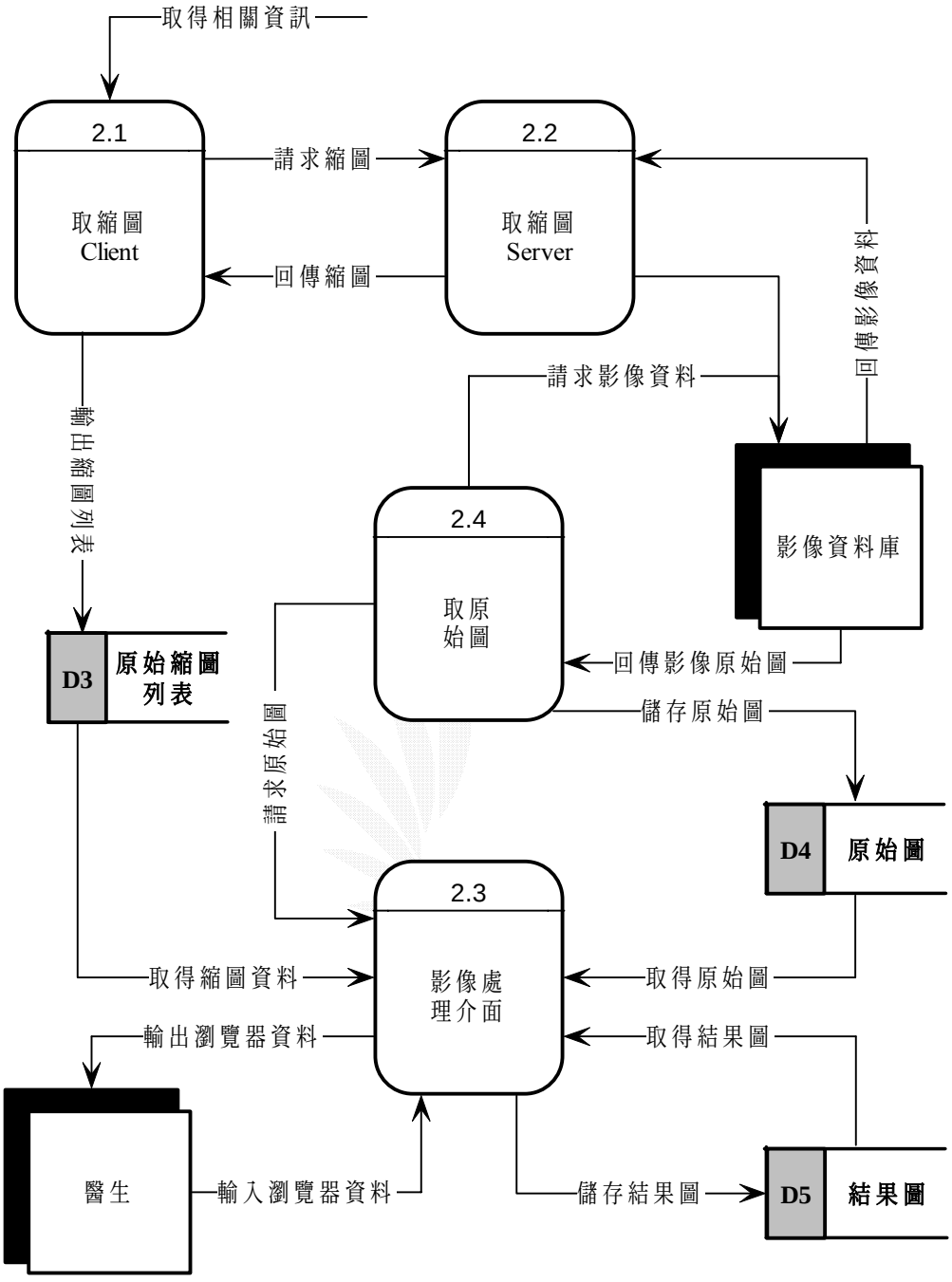


圖 二-3-4 DIAGRAM 2

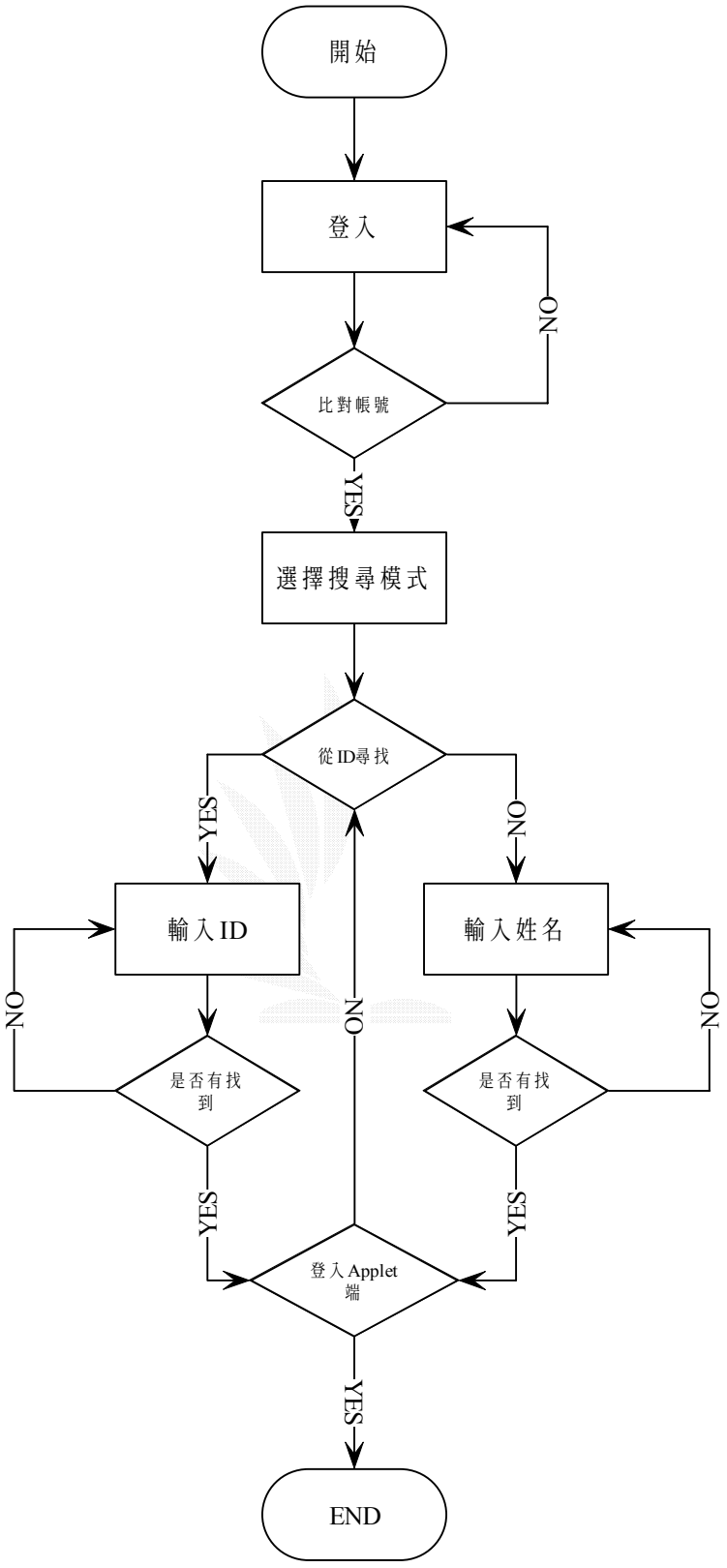

開始
登入
NO
比對帳號
YES
選擇搜尋模式
從ID尋找
YES
NO
輸入ID
輸入姓名
NO
NO
NO
是否有找到
是否有找到
YES
YES
登入Applet端
YES
END

圖 二-3-5 DF0

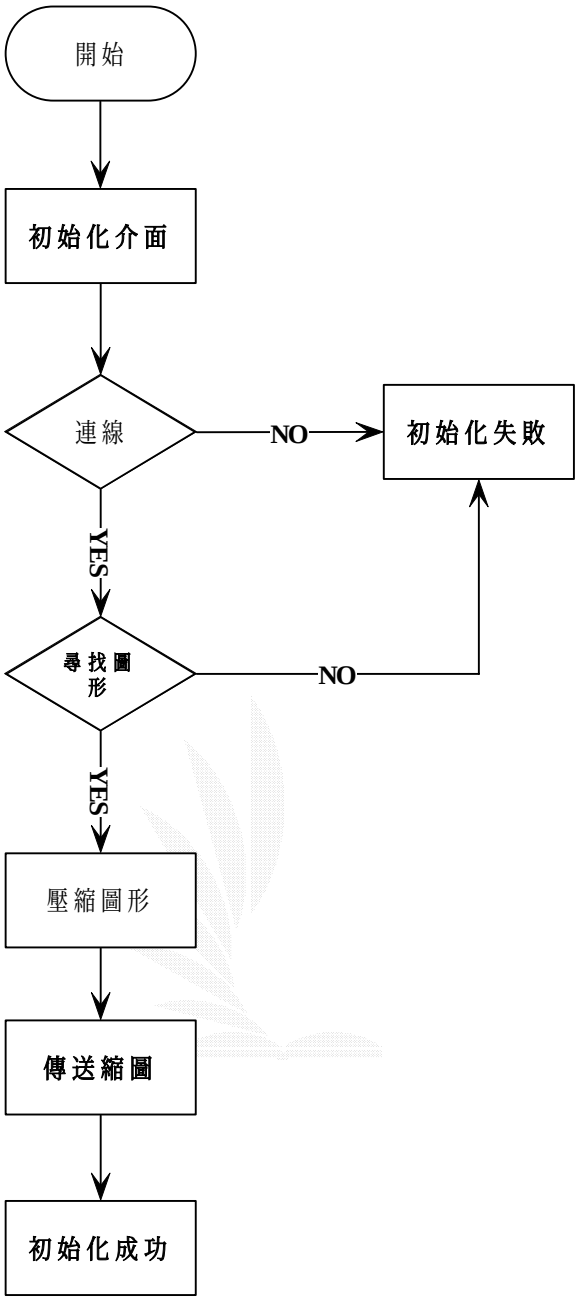


圖 二-3-6 DF1

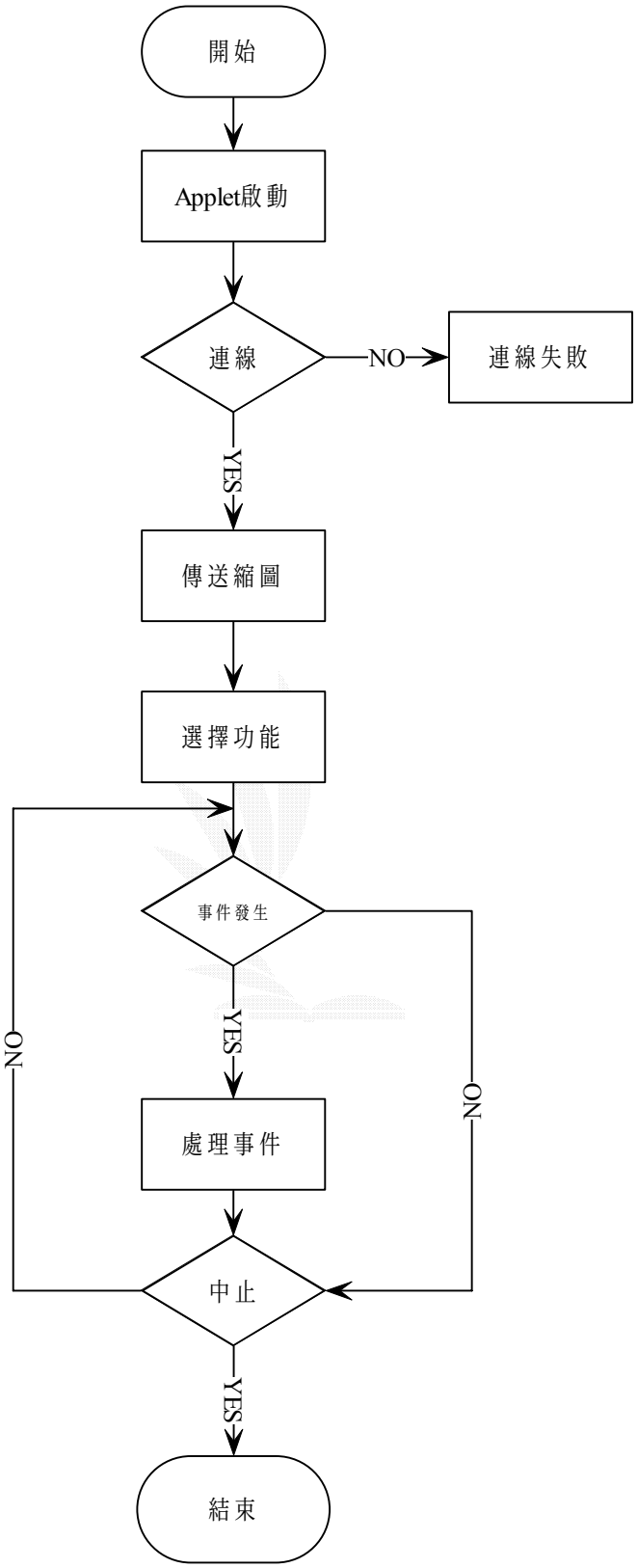

開始
Applet啟動
連線
NO
連線失敗
YES
傳送縮圖
選擇功能
事件發生
YES
NO
NO
處理事件
中止
YES
結束

圖 二-3-7 DF2

二-4　軟硬體配備

硬體配備建議：

在我們測試時發現、當 Applet 載入系統所需的元件時，記憶體的需求不少，且可能是演算法的效率不彰，運算某些功能讓 CPU 很吃力，所以建議 CPU 在 1G 以上，記憶體 512MB 以上

軟體配備：

支援可結合 JAVA VIRTUAL MACHINE 的瀏覽器

支援上述瀏覽器之 OS

JAVA VIRTUAL MACHINE 版本 1.4 以上

第三章　影像處理

三-1　點處理運算

點處理運算為一無記憶體的運算，而迴旋運算是一有記憶體的運算，執行迴旋運算時，過去的輸入需儲存起來以計算目前的輸出，點處理運算可分為函數轉換和算數邏輯運算。

函數轉換

函數轉換將輸入灰度 f 根據轉換函數 T 映設至 g。

$$g = T(f)$$

下面影像灰度值皆在(0,255)間。

二值化函數：

$$\text{(a)}\, T(f) = \begin{cases} 0 & f \le T_0 \\ 255 & f \ge T_0 \end{cases}$$

$$\text{(b)}\, T(f) = \begin{cases} 255 & a \le f \le b \\ 0 & f\text{在在其他範} \end{cases}$$

這些函數將影像中的物體跟背景分離，其轉換關係圖三-3-1，這些函數常應用在字元辨識、圖形辨識中，將物體跟背景分離。

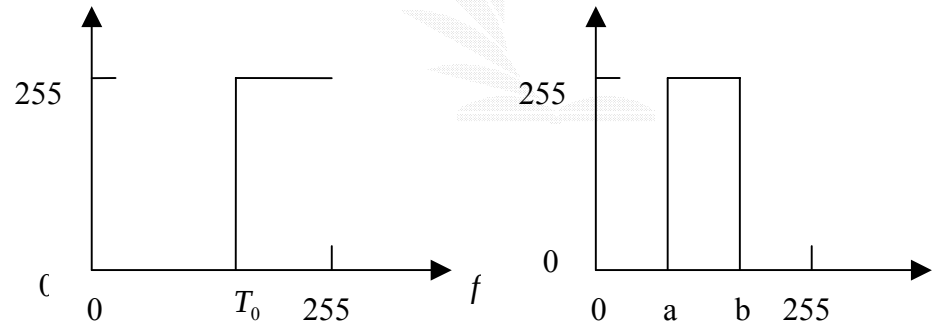


圖 三-1-1 影像二值化函數

反白函數：

$$T(f)255 - f$$

此函數將將影像中黑變白、白變黑，其函數如圖三-1-2

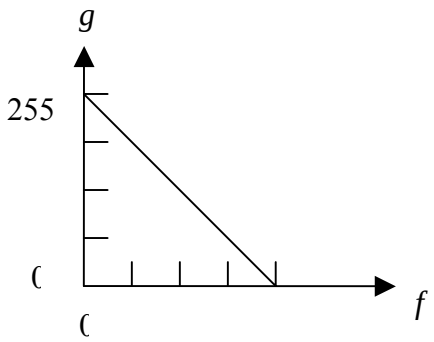


圖 三-1-2 反白函數

二補數函數：

$$T(f)=\begin{cases} f+128 & f<128 \\ f-128 & f\geq 128 \end{cases}$$

此函數將原影像由(0,255)轉換至(-128,127)之 2 補數表示，影像在運算過程中常會出現負數，一般固定位元運算時，需將影像轉換為 2 補數以避免溢位情形，2 補數的一個優點是只要結果不溢位，運算的加減過程中溢位情況可以忽略不管，其結果不會溢位，一般以硬體處理影像時，位元長度是固定不變的，此時就需以 2 補數表示影像，此轉換函數如圖三-1-3。

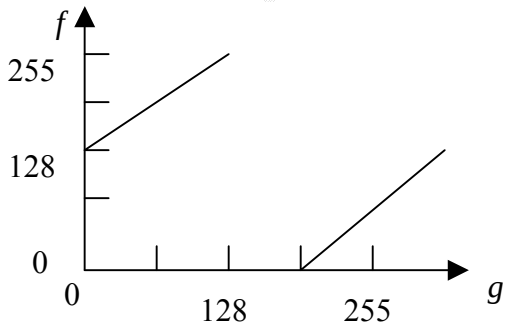


圖 三-1-3 2 補數影像轉換函數圖

量化函數：

8 位元影像灰度可以二進位表示成

$$f=b_7 2^7+b_6 2^6+\Lambda\ \Lambda+b_1 2^1+b_0$$

我們可將此影像重新量化，取最高 m+1 位元量化，將原影像量化至 2^{m+1} 灰度，

$$T(f) = b_7 2^7 + b_6 2^6 + \Lambda\ \Lambda\ b_{7-m} 2^{7-m}$$

$$= 2^{7-m} \bullet INT\left[f \Big/ 2^{7-m} \right] \qquad m=0\sim6 \qquad \text{式 三-1-1}$$

對比伸張強化函數：

一些看不清楚的影像可能是因為影像本身的動態範圍太窄，以致對比太弱，對比伸張函數可將動態範圍在(*min*,*max*)間的影像映射至最大的動態範圍(0,255)。

$T(f)=0 \qquad f<min$

$T(f)=255 \qquad f>max \qquad$ 式 三-1-2

$$T(f)=\frac{f-a}{b-a} \bullet 255$$

有時由於雜訊的影響，*min* 和 *max* 不是真正影像的最大和最小值，而是由雜訊造成的，此時(*min*,*max*)比真正影像動態範圍大，導致式三-1-1 的斜率不夠大，伸張的效果不好。解決這個問題，我們可設灰度 a 為累加點數小於某一百分比如 5%，b 為累加點數小於另一百分比如 98%的灰度，則式三-1-2 可修改為

$T(f)=0 \quad f<a$，$T(f)=255 \qquad f>b$

$$T(f)=\frac{f-a}{b-a} \bullet 255 \quad a \le f \le b \qquad \text{式 三-1-3}$$

如此即可得到更好的強化效果。實際上我們可將式三-1-2 或式三-1-3 修改得更有彈性，將影像灰度值切割成不同區間，每一區間以不同斜率強化。

三-2　均化濾波（Smoothing filter）

(1)介紹

均化濾波又稱作低通濾波，此濾波器會使信號變化變得較平緩，強化變化平緩的成份（低頻成份），抑制變化較快的成份（高頻成份）故又稱低通濾波器，通常罩遮範圍越廣濾波器就愈能依照其所定條件如低通濾波頻寬範圍而設計，而這裡將以 3×3 範圍為主。

(2)功用

此種濾波器主要用來去除雜訊，去除高頻信號成份和來重新取樣。

(3)證明

設影像 $f(x,y)$，含有雜訊 $n(x,y)$，不受干擾 $f'(x,y)$，所以 $f(x,y) = f'(x,y) + n(x,y)$。

$$f'(x,y) = \begin{matrix} 4 & 4 & 4 & 4 & 4 & 4 & 4 \\ 4 & 4 & 4 & 4 & 4 & 4 & 4 \\ 4 & 4 & 4 & 4 & 4 & 4 & 4 \\ 4 & 4 & 4 & 4 & 4 & 4 & 4 \\ 4 & 4 & 4 & 4 & 4 & 4 & 4 \\ 4 & 4 & 4 & 4 & 4 & 4 & 4 \\ 4 & 4 & 4 & 4 & 4 & 4 & 4 \end{matrix}$$

$$n(x,y) = \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 4 & 1 & 0 & 0 \\ 0 & 0 & 2 & 5 & 2 & 0 & 0 \\ 0 & 0 & 3 & 6 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

$$f(x,y)=\begin{matrix}4&4&4&4&4&4&4\\4&4&4&4&4&4&4\\4&4&5&8&11&4&4\\4&4&6&9&12&4&4\\4&4&7&10&13&4&4\\4&4&4&4&4&4&4\end{matrix}$$

經過 Smoothing filter(取四捨五入)⇒

$$f(x,y)=\begin{bmatrix}5&8&11\\6&9&12\\7&10&13\end{bmatrix}$$

$$f'(x,y)=\begin{bmatrix}4&4&4\\4&4&4\\4&4&4\end{bmatrix}$$

$$n(x,y)=\begin{bmatrix}1.3&1.6&1.3\\2.3&3&2.3\\1.8&2.3&1.8\end{bmatrix}$$

由此可知雜訊值可被降低

由上例可證：

$$f(x,y)=f'(x,y)+n(x,y)$$

取其 Smoothing filter

$$\Rightarrow g(x,y)=\frac{1}{9}\sum_{(k,1)\in w}\sum_{(k,1)\in w}f'(x-k,y-1)+\frac{1}{9}\sum_{(k,1)\in w}\sum_{(k,1)\in w}n(x-k,y-1)$$

設雜訊影像$n(x,y)$，方差$A\sigma^2$，則雜訊的信號方差：

$$Var\left[\frac{1}{9}\sum\sum n(x-k,y-1)\right]$$

$$=\left(\frac{1}{9}\right)^2 Var\left[\sum\sum n(x-k,y-1)\right]=\left(\frac{1}{9}\right)^2*\left(9\sigma^2\right)$$

$$=\frac{\sigma^2}{9}$$

其中

$$Var[x+y] = Var[x] + Var[y]$$
$$Var[kx] = k^2 Var[x]$$
$$Var[x] = E\left[(x-\bar{x})^2\right]$$

是故雜訊能量減少了 9 倍，但相對的 Smoothing filter 會抑制 image 的高頻成分，邊緣變化變得平滑，就好像模糊了影像一樣。

三-3　中值濾波 (median filter)

(1)介紹

1971 年，J.W.Jukey 提出中值濾波器來處理非線性訊號方法。它在某些條件下，可克服線性濾波器如平均濾波，均方濾波所引起影像系擷模糊的問題，並對於濾波脈衝干擾及影像掃描雜訊最有效。

(2)功用

1.對某些輸入訊號，中值濾波有不變性

在 $2n+1$ 視窗內單調增加或單調減少的某些輸入訊號，中值濾波之輸出訊號保持等於輸入訊號，即

$$f_{i-n} \le f_{i-n+1} \cdots \frac{\le}{>} f_i \frac{\le}{>} f_{i+n}$$

則 $\{y_i\}=\{f_i\}$ 式中 y_i 為中值，f_i 為資料序列。

表三-2-1 為二維視窗和與之對應的不變輸入圖形，它有與視究對頂角連線垂直邊緣線保持不變的特性，利用此特性，中值濾波可去除雜訊，又能保持影像中物體的邊。

表 三-3-1中值濾波常用視窗和對應的圖形

o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o o	o o o o o o o o o o o o o	o o o o o o o o o	o o o o o o o o o	o o o o o
(a)	(b)	(c)	(d)	(e)

2.中值濾波具消除雜訊功能

因中值濾波為非線性運算，因此對隨機輸入雜訊的分析很複。當輸入為均值雜訊時，中值濾波的雜訊方差 σ^2_{med} 約為

$$\sigma^2 = \frac{1}{4mf(\overline{m})} \overset{\bullet}{\sim} \frac{\sigma_i^2}{m + \frac{\pi}{2} - 1} \times \frac{\pi}{2}$$

式中：$\overline{m}$：輸入雜訊均值

$f(\overline{m})$：輸入雜訊密度函數

σ_i^2：輸入雜訊功率

$$\sigma_i^2 = \frac{1}{m}\sigma_i^2$$

對隨機雜訊的抑制能力，平均濾波比中值濾波好些。而對脈衝寬度索於$\frac{m}{2}$的窄脈衝干擾，中濾波有很好的效果。

3.中值濾波不變的頻譜特性

因為中值濾波不為線性運算，因此在顛率不存在對應關係，下列方法易於了解中值濾波輸入輸出頻譜變化的情況。

$$H \triangleq \left|\frac{G}{F}\right|$$

由理論及實驗證明，顛率響應 H 和輸入訊號頻譜 G 有關，並且是振動不大的曲線。如圖三-3-1 所示。

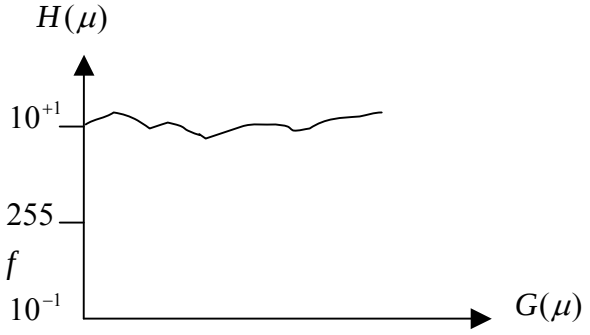
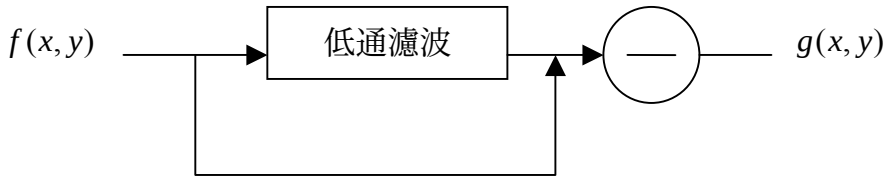


圖 三-3-1 H and G 的關係曲線

由圖中可知$H(\mu)$變化不大，可以認定訊號經中值濾波後，頻譜不變。

三-4　高通濾波（high pass filter）

高通濾波強化影像高頻的成份，濾去低頻成份，低通濾波濾去高頻成份，因此只要讓原影像和低通濾波的影像相減即可獲得高通濾波的結果，系統結構如圖三-2



(a)

$$\begin{bmatrix} \frac{-1}{9} & \frac{-1}{9} & \frac{-1}{9} \\ \frac{-1}{9} & \frac{8}{9} & \frac{-1}{9} \\ \frac{-1}{9} & \frac{-1}{9} & \frac{-1}{9} \end{bmatrix} \qquad \begin{bmatrix} 0 & \frac{-1}{8} & 0 \\ \frac{-1}{8} & \frac{1}{2} & \frac{-1}{8} \\ 0 & \frac{-1}{8} & 0 \end{bmatrix}$$

(b)

圖 三-4-1　(a) 高通濾波系統結構 (b) 3 × 3 高通濾波罩遮

設$W_{LP}(x,y)$表低通濾波罩遮，$W_{HP}(x,y)$表高通罩遮，則

$$W_{HP}(m,n) = \delta(m,n) - W_{LP}(m,n)$$

$$\delta(m,n) = \begin{cases} 1 & m = n = 0 \\ 0 & \text{其它} \end{cases} \qquad \text{式 三-4-1}$$

利用式 3-4.1 和低通濾波罩遮，可求出高通濾波罩遮如圖三-4-1(b)

注意，高通濾波將信號直流成份濾除，因此高通濾波的結果信號為交流信號有正有負，而一般影像的顯示皆為正值，故須將輸出影像取絕對值或適當平移。

實現一個高通（銳化）空間濾波器所需要的沖激響應的形狀指出，濾波器在它的中心附近應當具有正的係數，在外部周圍具有負的係數。對於一個 3×3 遮罩，選擇中心位置上為正係數，其餘位置上為負係數，這種遮罩滿足上述條件。

$$R = w_1 z_1 + w_2 z_2 + \mathrm{K} + w_9 z_9$$

$$\frac{1}{9} \times \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

這個濾波器消除零頻率項，該項的消除把影像的平均灰值度簡化為 0，顯著的降低了影像的整體對比度。原始影像由細節和灰度慢變化的背景組成，而結果是增強了在一個相單暗的背景中的邊緣。

把一幅影像的平均值減少為 0 意味著影像必須具有某些負灰階。因為我們只處理正值，高通濾波的結果包括某些比例和（或）截幅的形成，使得最後的灰階落在範圍 [0,L-1] 。取濾波後影像的絕對值使得所有值為正，通常不是一個好的想法，因為負值明顯地出現在影像中。

經過兩次微分後，參照微分強化章，我們使用 Laplace mask，所以可得

$$g(x,y) = \begin{cases} f(x,y) - \nabla^2 f(x,y) \\ f(x,y) + \nabla^2 f(x,y) \end{cases}$$ so we can show this mask

$$\begin{cases} \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix} \\ \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \end{cases}$$

然後經過一個叫 unsharp masking 的過程，可以表達出

$f_s(x,y) = f(x,y) - F(x,y)$

$f_s(x,y)$ denotes the sharpened image obtained by unsharp masking

$F(x,y)$ is a blurred version of $f(x,y)$

所以可推出 $f_{hb}(x,y) = Af(x,y) - F(x,y)$ where $A \geq 1$

$f_{hb}(x,y)$ high-boost filter

然後可在寫 $f_{hb}(x,y) = (A-1)f(x,y) + f(x,y) - F(x,y) = (A-1)f(x,y) + f_s(x,y)$

是故 $f_{hb}(x,y)=\begin{cases}Af(x,y)-\nabla^2 f(x,y)\\ Af(x,y)+\nabla^2 f(x,y)\end{cases}$ so we can show this mask is

$\begin{bmatrix}-1 & -1 & -1\\ -1 & A+8 & -1\\ -1 & -1 & -1\end{bmatrix}$ or $\begin{bmatrix}0 & -1 & 0\\ -1 & A+4 & -1\\ 0 & -1 & 0\end{bmatrix}$ $f_{hb}(x,y)$ high-boost filter

承上，由於已知 $f_{hp}(x,y)=f(x,y)-f_{lp}(x,y)$

$f_{hp}(x,y)$ high-pass filter

$f_{lp}(x,y)$ low filter

and

$$f_{hp}(x,y)=Af(x,y)-f_{lp}(x,y)=(A-1)f(x,y)+f(x,y)-f_{lp}(x,y)=(A-1)f(x,y)+f_{hp}(x,y)$$

where $A\geq 1$

so we can write

$$\left(f_{hb}(x,y)\middle/f(x,y)\right)=(A-1)+\left(f_{hp}(x,y)\middle/f(x,y)\right)=H_{hb}(x,y)=(A-1)+H_{hp}(x,y)$$

sometimes it is advantageous to accentuate the contribution to enhancement made by the high frequency components of an image. In this case, we simply multiply a high pass filter function by a constant and add offect so that the zero frequency term is not eliminated by the filter.

$H_{hfe}(x,y)=a+bH_{hp}(x,y)$ where $a\geq 0$, $b>a$

一般而言 $0.25\leq a\leq 0.5$, $1.5\leq b\leq 2.0$

$H_{hfe}(x,y)$ high-frequency emphasis

(2)功用

銳化的主要目的是增強影像中的細微部份或增強已經模糊了的細節。模糊原因可能是影像獲取的特定方法的誤差，也可以是方法的自然效應。影像銳化的使用各式各樣，包括的應用範圍從電子印刷和醫學影像到工業檢驗和現代化武器的自動目標偵測。

三-5　時間平均濾波（temporal filter）

(1)介紹

如果雜訊隨著時間而改變，且每一時間的雜訊彼此皆不相關，我們可以時間平均的方式達到降低雜訊能量的效果，同時維持空間的解析度。考慮一幅雜訊影像 $g(x,y)$，它是雜訊 $\eta(x,y)$ 疊加在一幅原始影像 $f(x,y)$ 上面形成的，即

$$g(x,y) = f(x,y) + \eta(x,y)$$

這裡假定在每一對座標 (x,y) 上雜訊互不相關且具有零平均值。

(2)功用

其方法的目的是透過一組雜訊影像 $\{g(x,y)\}$ 的相加來減少雜訊效應。

(3)證明

如果雜訊滿足上述條件，我們可很簡單低證明如下結論：如果一幅影像是由 M 幅不同的雜訊影像經過平均而形成的，即

$$\bar{g}(x,y) = \frac{1}{M}\sum_{i=1}^{M} g_i(x,y)$$

則有

$$E\{\bar{g}(x,y)\} = f(x,y)$$

和

$$\sigma^2_{g(x,y)} = \frac{1}{M}\sigma^2_{\eta(x,y)}$$

這裡 $E\{\bar{g}(x,y)\}$ 是 $\bar{g}$ 的期望值，$\sigma^2_{g(x,y)}$ 和 $\sigma^2_{\eta(x,y)}$ 是所有座標 (x,y) 上 $\bar{g}$ 和 η 的方差。平均影像中任何一點上標準差為

$$\sigma_{g(x,y)} = \frac{1}{\sqrt{M}}\sigma_{\eta(x,y)}$$

三-6　局部平均變異強化 (local mean variance enhancement)

(1)介紹

長條圖強化影像是利用整體的資訊來強化影像，使整張影像不偏黑或偏暗且對比最強烈，但此種方法對於局部影像偏黑或偏暗則無法得到強化的效果，如逆光攝影影像，整張影像對比很好但逆光的部份卻偏暗，長條圖強化會將整張影像動態範圍（對比）拉開，由於整張影像對比已經很好，對於局部偏暗的部份不能得到強化的效果。一般而言對比愈強的影像其變異數也愈大，局部平均變異強化乃利用此特點一方面維持局部的平均值（直流成份）同時加強此影像的交流部份，如果局部的變異數偏低就將它放大，因為影像交流的部份一般代表影像中的紋路變化，如此即可強化影像中局部偏暗或偏亮部份。局部平均變異強化公式如下：

$$m(x,y)=\frac{1}{N_w}\sum_{(x,j)\in w} f(x-i,y-i) \qquad \text{式 三-6-1}$$

$$\sigma(x,y)=\sqrt{\frac{1}{N_w}\sum_{(i,j)\in w}\left|f(x,y)-m(x,y)\right|^2} \qquad \text{式 三-6-2}$$

N_W 代表局部範圍 W 的點數，$m(x,y)$ 代表影像 $f(x,y)$ 的局部平均影像，σ(x,y) 是 $f(x,y)$ 的局部變異數，由式三-6.2 可知 $\sigma(x,y)$ 愈大的點其灰度差異較大，對於點 $f(x,y)$，如果 $\sigma(x,y)$ 愈小則將差異的部份 $f(x,y)-m(x,y)$ 以和 $\sigma(x,y)$ 成反比的倍數放大，注意在執行式三-6.1 要注意運算中的溢位，$k/\sigma(x,y)$ 的值應限制在某種範圍內。

$$g(x,y)=A(x,y)\left[f(x,y)-m(x,y)\right]+m(x,y)$$

其中

$$A(x,y) = k^{\frac{M}{\sigma(x,y)}} \qquad 0 < k < 1$$

變數 A，m 和 σ 的數值與 (x,y) 的鄰域定義有關。作用於 $f(x,y)$ 與局部均值 $m(x,y)$ 之差的局部增益因子 $A(x,y)$ 將局部方差放大。因為 $A(x,y)$ 與灰度的標準差成反比，所以低對比度的區域具有高增益。加在式四後面的平均值用於恢復局部區域上影像的平均灰度。實際中，為了抵銷各個區域內大的灰階偏差。常常希望後加一個局部平均值因子並且將 $A(x,y)$ 限制在 $(A_{\min}, A_{\max})$ 之間。

灰度平均值和方差（或標準差）就是常被使用的兩種特性，因為它們與影像的面貌有關。就是說，平均值是平均亮度的一個量測，方差是對比度的量測。以這些概念為基礎的一種典型局部轉換在每個像素位置 (x,y) 上使用如下公式把輸入影像的灰度 $f(x,y)$ 映射為一幅新的影像 $g(x,y)$

在上述轉換中，$m(x,y)$ 和 σ 分別是在以 (x,y) 為中心的鄰域上計算出的灰階平均值和標準差，M 是 $f(x,y)$ 的整體平均，k 是一個常數，其取值範圍為：$0 \le k \le 1$。

利用同樣的原理，我們也可以局部長條圖強化法達到局部強化的效果，對於每一點 $f(x,y)$ 我們算出其周圍 N_W 點如 7×7 或 9×9 或 13×13 的範圍的長條圖，利用此長條圖的等化曲線來強化點 $f(x,y)$，如此每一點皆可能被對應到(0.255)範圍內的灰度值。本程式用 3×3 矩陣。

(2)功用

局部平均變異強化

三-7　長條圖分布 (histogram equalization or histogram linearization)

(一)介紹

利用長條圖強化

對比伸長強化的缺點是需要經由錯誤嘗試,交替式的處理才能得到一較好的強化效果，是否能夠自動的強化影像呢？這裡我們介紹一種利用長條圖強化的方式稱為長條圖等化法，此方法的目的是希望強化後的影像能達到全動態範圍且灰度值分布均勻。根據機率論的變數變換定理，取任意機率的分佈函數為變數變換公式，能將機率函數轉換為在 (0,1) 間的平均分配機率密度函數。應用此種原理即可將原長條圖轉換為均勻分佈的長條圖。

設原機率密度函數為 $p(f)$，則其分佈函數：

$$T(f)=\int_0^f p(x)dx \qquad 0\le x\le 1$$

讓 $s=T(f)$ 表示轉換後影像的灰度變數，則

$$p(s)=p(f)\frac{df}{ds}=p(f)*\frac{1}{p(f)}=1$$

故變數 s 的機率分佈為均勻分佈。

由前面推倒可之長條圖強化函數可表示如下：

$$s=T(f)=\int_0^f p(x)dx \qquad \text{式 三-7-1}$$

式三-7-1 為理論的結果，但實際上因影像灰度值為有限個能階，式三-7-1 所計算的 s 值並不一定在次有限能階上，此時須將 s 重新量化，因此實際並不能得到一灰度均勻分佈的結果。

直方圖修正

直方圖修正的工作主要是針對影像灰度曝光不夠均勻所引起的問題，例如灰度集中在高度範圍，則照片曝光；集中在低亮度範圍，則照片曝光不足，兩者都使照片不清楚。如圖三-7-1

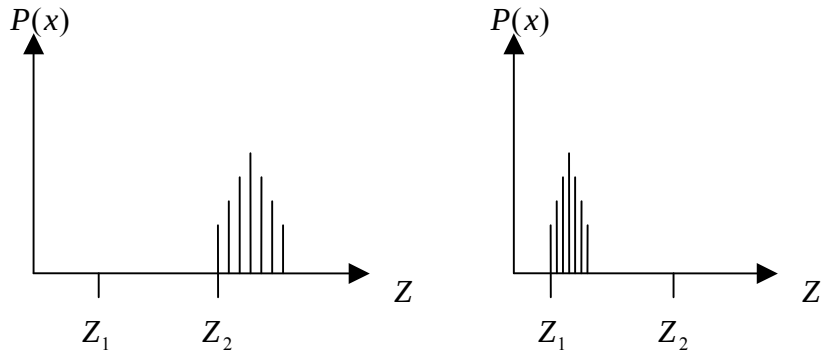


圖 三-7-1 以模組化結構執行局部平均變異強化

爲了將直方圖標準化，假定變數 r 代表待加強影像的灰階且灰階為連續的變化，其範圍在 $[0,1]$ 之間，$r=0$ 對應於黑，$r=1$ 對應於白。設灰階度 z 標準化為 r，變換後的影像任一個灰度級 z' 標準化為 s，s 應滿足下列條件：

1. $T(r)$ 在 $0<r<1$ 區間是單直且單調增加。

2. 對於 $0<r<1$，$0<T(r)<1$。

其中

$$s=T(r)$$

為原始影像每一像素灰階 r 產生灰階 s 的轉換式。

上述條件 1. 為使灰階從黑到白的次序 2. 使映射轉換後的影像灰階在允許的範圍內。圖三-4 為這條件的轉換函式。

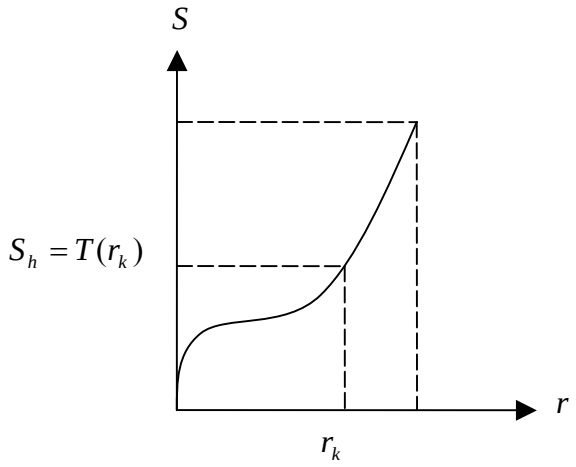


圖 三-7-2 灰階轉換函數

從 s 到 r 的反轉換如下形式：

$$r = T^{-1}(s) \qquad 0 \le s \le 1$$

其中 $T^{-1}(s)$ 也滿足上面的條件 1.、2.。

若影像灰階屬於連續量，則開始灰階 $p_r(r)$ 和轉換後灰階 $p_s(s)$ 分別表示出現的機率密度函數。由機率理論知，如果 $p_r(r)$ 和 $T_r(r)$ 為已知，且 $T^{-1}(s)$ 滿足條件 1.，則轉換後密度函數為

$$p_s(s) = \left[p_s(r) \frac{dr}{ds} \right]_{r=T^{-1}(s)} \qquad \text{式 三-7-2}$$

考慮轉換函數

$$s = \int ds = \int_0^r p_r(w) dw \qquad 0 \le r \le 1 \qquad \text{式 三-7-3}$$

式中，w 為虛設積分變數，利用 $T_r(r)$ 去控制影像灰階的機率密度函數，進而改善影像，上式右邊為 r 的累積分佈函數，它必須滿足條件 1.、2.。

從式三-7-3 得到

$$\frac{ds}{dr} = p_r(s)$$

將它代入式 3-7-2

由上面可知 s 變換後之機率密度式均勻的：若以增強觀點而言，奇意指它增強了像素的動態範圍，對於影像的外觀有極佳的效果。例如有一幅影像其灰度機率密度函數 $p_r(r)$ 為

$$p_r(r) = \begin{cases} -3r+3 & 0 \le r \le 1 \\ 0 & \textit{其他} \end{cases}$$

代入式三-7-3

$$\begin{aligned} s = T(s) &= \int_0^r p_r(w)dw \\ &= \int_0^r (-3r+3)dr \\ &= -\frac{3}{2}r^2 + 3r\Big|_0^r \\ &= -\frac{3}{2}r^2 + 3r \end{aligned}$$

現在驗證 $T_r(r)$ 變換後的影像灰度級分佈是否均勻，亦即 $p_s(s)$ 是否為 1。

而 r 之範圍在 $[0,1]$ 之間，負號不成立。

可知 $T(r) = -\frac{3}{2}r^2 + 3r$ ，轉換後的影像其灰度分佈式均勻的，如圖三-7-3 所示。

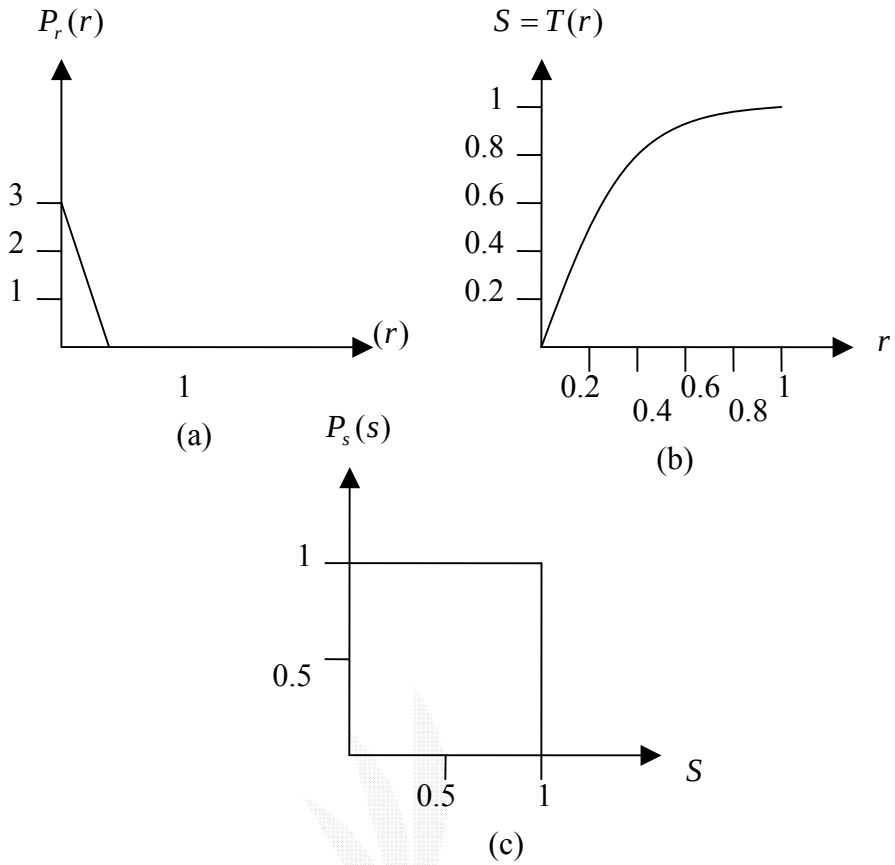


圖 三-7-3 直方圖均勻化

爲了對於數位影像的處理，將上面所述公式離散化，與機率形式：

$$p_r(r_k) = \frac{n_k}{n} \qquad 0 \le r_n \le 1 \qquad k = 0,1,\Lambda\ L-1$$

其中 L 是灰階數目，$p_r(r_k)$ 市第 k 個灰街的機率，n_k 是灰階 r_k 在影像中出現的次數，n 是像素總數。$p_r(r_k)$ 和 r_k 之關係圖叫直方圖，獲致均勻直方圖的技術稱為直方圖均勻化（histogram equalization）或直方圖線性化（histogram linearization）。

$$s_k = T(r_k) = \sum_{j=1}^{k} \frac{n_j}{n} = \sum_{k=0}^{h} p_r(r_k) \quad 0 \le r_k \le 1 \quad h = 0,1,\Lambda\ ,L-1$$

反轉換式　$r_k = T^{-1}(s_k)$　　$0 < s_k < 1$

正反轉換式都滿足條件 1.、2.。

例：假如有一幅影像，具有 64 × 64 個像點，8 個灰階度，其機率分佈如下表三-7-1，現將此影像以直方圖均勻化。

表 三-7-1 影像以直方圖均勻化

r_k	n_k	$p_r(r_k)=\frac{n_k}{n}$
$r_0=0$	790	0.19
$r_1=\frac{1}{7}$	1023	0.25
$r_2=\frac{2}{7}$	850	0.21
$r_3=\frac{3}{7}$	656	0.16
$r_4=\frac{4}{7}$	329	0.08
$r_5=\frac{5}{7}$	245	0.06
$r_6=\frac{6}{7}$	122	0.03
$r_7=1$	81	0.02

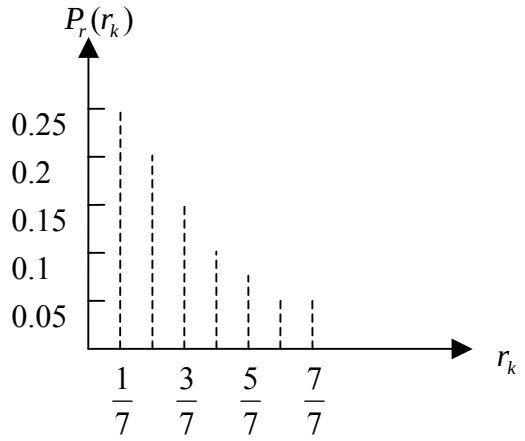
表 三-7-2 直方圖均勻化之過程

原灰度級	變換函數 $T(r_k)$ 值	像點數	量化級	新灰度級	新灰度級分佈
$r_0 = 0$	$T(r_0) = s_0 = 0.19$	790	0		0
$r_1 = \frac{1}{7}$	$T(r_1) = s_1 = 0.44$	1023	$\frac{1}{7} = 0.14$	$\rightarrow s_0'(790)$	$\frac{790}{4096} = 0.19$
$r_2 = \frac{2}{7}$	$T(r_0) = s_0 = 0.19$	850	$\frac{2}{7} = 0.29$		
$r_3 = \frac{3}{7}$	$T(r_3) = s_3 = 0.81$	656	$\frac{3}{7} = 0.43$	$\rightarrow s_1'(1023)$	$\frac{1023}{4096} = 0.25$
$r_4 = \frac{4}{7}$	$T(r_4) = s_4 = 0.89$	329	$\frac{4}{7} = 0.57$		
$r_5 = \frac{5}{7}$	$T(r_5) = s_5 = 0.95$	245	$\frac{5}{7} = 0.71$	$\rightarrow s_2'(850)$	$\frac{850}{4096} = 0.21$
$r_6 = \frac{6}{7}$	$T(r_6) = s_6 = 0.98$	122	$\frac{6}{7} = 0.86$	$\rightarrow s_3'(985)$	$\frac{985}{4096} = 0.24$
$r_7 = 1$	$T(r_7) = s_7 = 1$	81	1	$\rightarrow s_4'(448)$	$\frac{448}{4096} = 0.11$

一段數學式子

依圖樣方法，計算 s2，s3~s7 可得值如下：

s2=0.65，s3=0.81，s4=0.89，s8=0.95，s6=0.98，s7=1



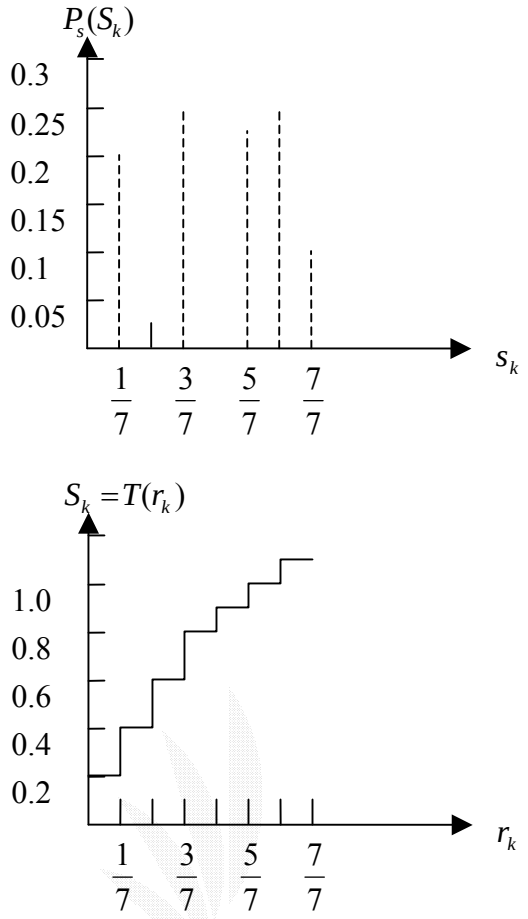


圖 三-7-4

將 $p_r(r_k)$， $p_s(s_k)$ 和 $p_k(T_{rk})$ 分別作圖，如圖三-7-4 所示。雖然影像給定之 r_k 值是等間隔的（均勻量化），而經由 $s_k = T(r_k)$ 轉換後，所得之 s_k 不為等間隔的，為了得到和原影像等間隔之均勻量化，對每一 s_k 取靠近之量化值，重新量化之心灰度級 s0’、s1’、s2’、s3’、s4’，再由此數值得到其灰度級分佈。

轉換後影像直方圖並非十分均勻，這是由於重新量化時，將每一個 r 灰度級之像點作一整體數，經過這樣的灰度轉換由 7 級減為 5 級，使含有像點數的灰度級間隔拉大了。像點數少的灰度級之間隔壓縮了，使可接收的視覺量增加了。

直方圖一般修正方法

爲了要得到較高的影像對比度，亦即增強影像，可採用直方圖均勻化處理方法，但是它使影像灰度級密集區佔有更多灰級度，而稀疏區減少灰度，完全利用各個密度區，而讓影像之對比度之效果更增強，

但如此影像並一定適合觀察，因此必須加以修正，方法有二，其一是用一個規定的機率密度函數來表示已要的直方圖。如表三-7-3 一般先將函數數位化，再變為直方圖。

表 三-7-3

	修正後要求的機率密度函數 $p_s(s)$	轉換函數 $s=T(r)$
指數分佈	$p_s(s)=\alpha\exp[-\alpha(s-s_{\min})]$ $s\geq s_{\min}$	$s=s_{\min}-\frac{1}{\alpha}\ln[1-p_r(r)]$
均勻分佈	$p_s(s)=\frac{1}{s_{\max}-s_{\min}}$ $s_{\min}\leq s\leq s_{\max}$	$s=[s_{\max}-s_{\min}]p_r(r)+s_{\min}$
雙曲線	$p_s(s)=\frac{1}{s[\ln(s_{\min}-\ln(s_{\min}))]}$	$s=s_{\min}\left[\frac{s_{\max}}{s_{\min}}\right]^{p+1}$
雷利分佈	$p_s(s)=\frac{s-s_{\max}}{\alpha^2}\exp\left\{-\frac{(s-s_{\min})^2}{2\alpha^2}\right\}$ $s\geq s_{\min}$	$s=s_{\min}+\left[2\alpha^2\ln\left(\frac{1}{1-p_r(r)}\right)\right]^{1/2}$

方法之二是用可控制直線段來完成直方圖，以得到所希望之形狀並加以數位標準化。

下面用一規定之機率密度函數表示已要直方圖的數值計算方法。假設 $p_r(r)$ 為原始影像之灰度機率密度函數，而 $p_z(z)$ 為希望影像之灰度機率密度函數，則對原影像均勻化處理。

由

$$s=T(s)=\int_0^r p_r(w)dw$$

對希望的影像進行均勻化處理

$$u=R(z)=\int_0^z p_z(w)dw$$

既然兩影像都均勻化處理，其灰度機率密度函數 $p_s(s)$ 和 $p_u(u)$ 相同，且都為 1，使用

$$z = R^{-1}(u) = R^{-1}(s) = R^{-1}[T(r)]$$

在數位影像中正反轉換其像點灰度值一一對應映射，而在連續影像中要探討解析式是否存在。

(二)功用

影像顏色過於偏向某色系時能與以擴散使顏色較均勻分佈進而減少過白過黑之類情形。

三-8　微分濾波 (Derivation filter)

(一)作用

微分濾波常應用於影像邊緣化或邊緣特徵抽取，利用影像的邊緣來表示影像的優點就是所需處理的資料大為減少，故微分濾波常被應用於影像的辨識。

一次微分對於邊緣過渡叫窄的影像效果較好，如果邊緣變化較緩，可利用二次微分幫忙偵測邊緣。

以下列出一些常用的微分濾波器。其中 Sobel 濾波器是由微分運算和低通運算所結合，固有降低雜訊的效果。

$$\begin{bmatrix} -1 & +1 \\ 0 & 0 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 \\ +1 & 0 \end{bmatrix}$$

Vertical & Horizontal

$$\begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix} \quad \begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix}$$

Robert Operators

$$\begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

Sobel Operators

以下列出一些影像二次微分所常用的拉普拉氏（Laplaction）運算：

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & +4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & +8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

$$\begin{bmatrix} +1 & -2 & +1 \\ -2 & +4 & -2 \\ +1 & -2 & +1 \end{bmatrix}$$

比較:

一次微分：

$\begin{bmatrix} 0 & 0 & 0 \\ -1 & +1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ 0 & +1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
Vertical	Horizontal
$\begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$	$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & -1 \end{bmatrix}$
SobelV	SobelH
$\begin{bmatrix} -1 & 0 & 0 \\ 0 & +1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ +1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
Robert1	Robert2

$\sqrt{\lvert Vertical\rvert^2 + \lvert Horizontal\rvert^2}$
Combined V &H
$\sqrt{\lvert SobelV\rvert^2 + \lvert SobelH\rvert^2}$
Sobel
$\sqrt{\lvert Robert1\rvert^2 + \lvert Robert2\rvert^2}$
Robert

二次微分：

表 三-8-1三種拉普拉式濾波器

$\begin{bmatrix} -1 & -1 & -1 \\ -1 & +8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & +4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} +1 & -2 & +1 \\ -2 & +4 & -2 \\ +1 & -2 & +1 \end{bmatrix}$
Lap1	Lap2	Lap3

(二)功用

1.邊緣檢測:

邊緣檢測是目前為止用來檢測有意義的不連續特徵的最常見方法，這是因為孤立的點和細線在大多數的實際應用中是較少出現的。

2.對比增強處理:

為了使模糊的影像變得更加清楚而採用對比增強，而影像模糊之主要原因乃是對其做平均或積分運算，因此若做反運算就使其變清晰。

(三)證明

由於影像是二微信號，一次微分有二個分量，此一向量稱為梯度（gradient），定義如下：

$$G[f(x,y)]=\begin{bmatrix}\frac{\partial f}{\partial x}\\ \frac{\partial f}{\partial y}\end{bmatrix}$$

我們取此項量的振幅 M（Magnitude）來表示邊緣，取此向量的方向 θ 來做進一步的分析如邊緣連接等，它們的定義為

$$M=\sqrt{(\frac{\partial f}{\partial x})^2+(\frac{\partial f}{\partial y})^2}$$

$$\theta=\tan^{-1}\frac{\partial f/\partial y}{\partial f/\partial x}$$

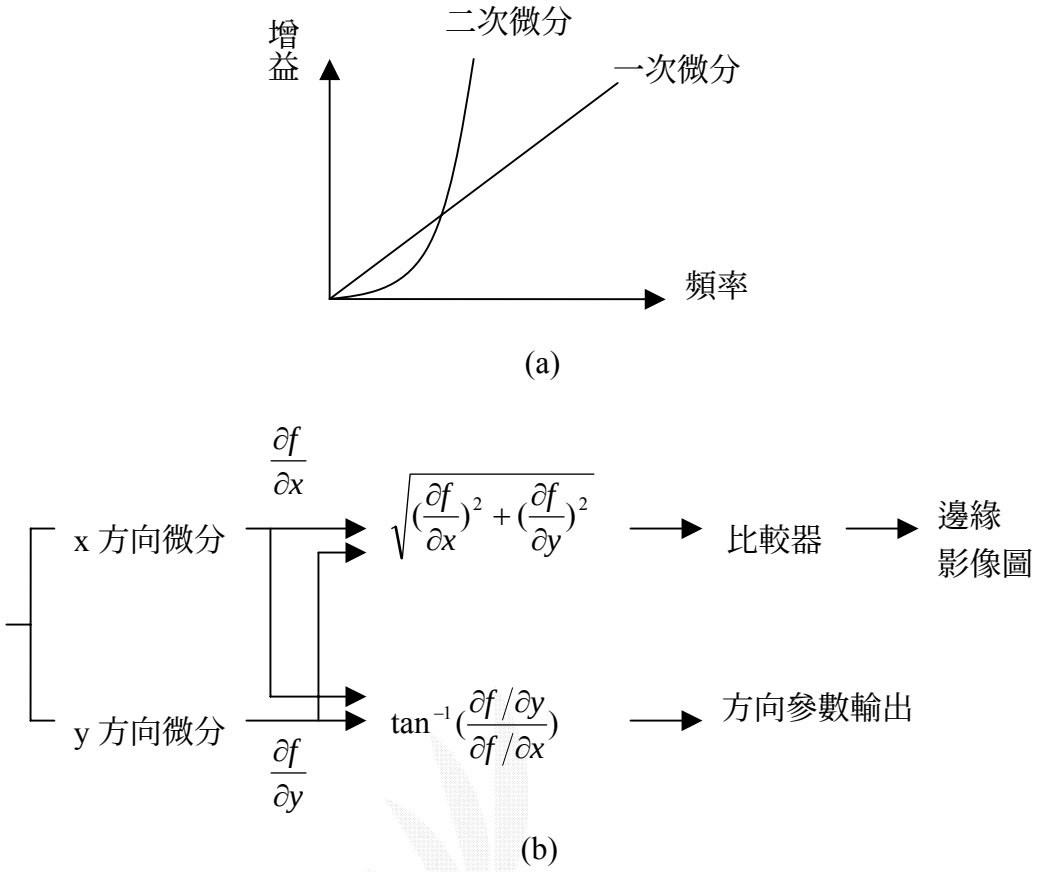


圖 三-8-1影像微分(a).頻譜 (b).系統流程圖

對於數位影像，我們可以相鄰點的差近似(式三-8.1)，且微分的方向可取任意兩個正交的方向進行微分，而梯度振幅的運算可取絕對值近似以減少運算，加快處理速度即

$$M = \left|\frac{\partial f}{\partial x}\right| + \left|\frac{\partial f}{\partial y}\right| \qquad \text{式 三-8-1}$$

$\frac{\partial f}{\partial x}$ 可以 $\frac{f(x+\Delta x, y) - f(x,y)}{\Delta x}$ 近似，取 $\Delta x = 1$ ，

則 $\frac{\partial f}{\partial x}$ 可以 $f(x+1,y) - f(x,y)$ 近似，同理 $\frac{\partial f}{\partial y}$ 可以 $f(x,y+1) - f(x,y)$ 近似之。$f(x+1,y) - f(x,y)$ 和 $f(x,y+1) - f(x,y)$ 可以遮罩運算表示如下：

$$f(x+1,y)-f(x,y)=f(x,y)*\begin{bmatrix}-1 & +1\\ 0 & 0\end{bmatrix}$$

$$f(x,y+1)-f(x,y)=f(x,y)*\begin{bmatrix}-1 & 0\\ +1 & 0\end{bmatrix}$$

此種近似為最直接的近似方式，經由一次微分後，影像中斜率為常數（一次變化）的區域其輸出微常數（大小和斜率成正比），影像中斜率為零的區域其輸出為零，舉例如下，左邊為原影像，右邊為 x 方向一次微分的結果，我們只計算方形圈住點的輸出

1	1	2	3	4	5	5
1	$\overline{2}$	$\overline{3}$	4	5	5	5
2	3	4	5	5	5	5
3	4	5	5	$\overline{5}$	5	5
4	5	5	5	5	5	5
5	5	5	5	5	5	5

x	*x*	*x*	*x*	*x*	*x*
x	1	1	*x*	*x*	*x*
x	*x*	*x*	*x*	*x*	*x*
x	*x*	*x*	*x*	0	*x*
x					

利用式三-8.1 可計算出影像梯度振幅，舉例如下，左邊為一 9×9 影像右邊為期梯度振幅輸出，注意方形的左上角有一缺口。

3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3
3	3	4	4	4	4	4	3	3
3	3	4	6	6	6	4	3	3
3	3	4	6	6	6	4	3	3
3	3	4	6	6	$\overline{6}$	4	3	3
3	3	4	4	4	4	4	3	3
3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3

0	0	0	0	0	0	0	0	*x*
0	0	1	1	1	1	1	0	*x*
0	1	0	2	2	2	1	0	*x*
0	1	2	0	0	2	1	0	*x*
0	1	2	0	0	2	1	0	*x*
0	1	2	2	2	4	1	0	*x*
0	1	1	1	1	1	2	0	*x*
0	0	0	0	0	0	0	0	*x*
x	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*

表 3-8.1 列出一些常用的微分運算器，*w*1 表某一方向的微分遮罩，*w*2 表和 *w*1 垂直的微分遮罩，以方塊圈住的係數表運算所對應的像素。其中 Prewitt 和 Sobel 運算有降低雜訊的效果，它們其實是微分運算和低通運算的結合，如 Sobel 的 *w*1 微分運算可表示成：

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

表 三-8-2 一些常用的微分運算器

	w_1	w_2
Robert	$\begin{bmatrix} -1 & +1 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 \\ 0 & +1 \end{bmatrix}$
	$\begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$
Prewitt	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$
Sobel	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$

$[-1 \quad 0 \quad 1]$為水平微分， $[-1 \quad 2 \quad 1]^T$ 為低通濾波。

一種取梯度的近似方法為取各種方向的微分運算中的最大值為其梯度，下面為以反時鐘方向一次旋轉 45°所得到的微分罩遮，此種方法可顧慮到每個方向的梯度，如前例左上角的缺口將產生。

$$w_1 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \quad w_2 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -1 \end{bmatrix} \quad w_3 = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

$$w_4 = \begin{bmatrix} 0 & -1 & -1 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{bmatrix} \quad w_5 = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad w_6 = \begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$w_7 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad w_8 = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}$$

取梯度輸出

$$g(x,y) = \max_{k=1,8}\{|g_k g(x,y)|\}$$
$$g_k(x,y) = f(x,y) * w_k \qquad k = 1, 2, \ldots, 8$$

注意：由於罩遮並未經過適當的調整，運算的值也許會超出(0,255)的範圍，一般如取罩遮係數絕對值的和當作除數，可保證運算的結果不會溢位。大家可以前面 9×9 影像比較不同梯度運算的效果。

一次微分對於邊緣過渡帶較窄的影像效果比較好，如果邊緣變化較緩，可以利用二次微分幫忙偵測邊緣。圖三-8-2 說明一次微分和二次微分的差異，利用二次微分我們可以偵測出此邊緣的外側和內側或中心。

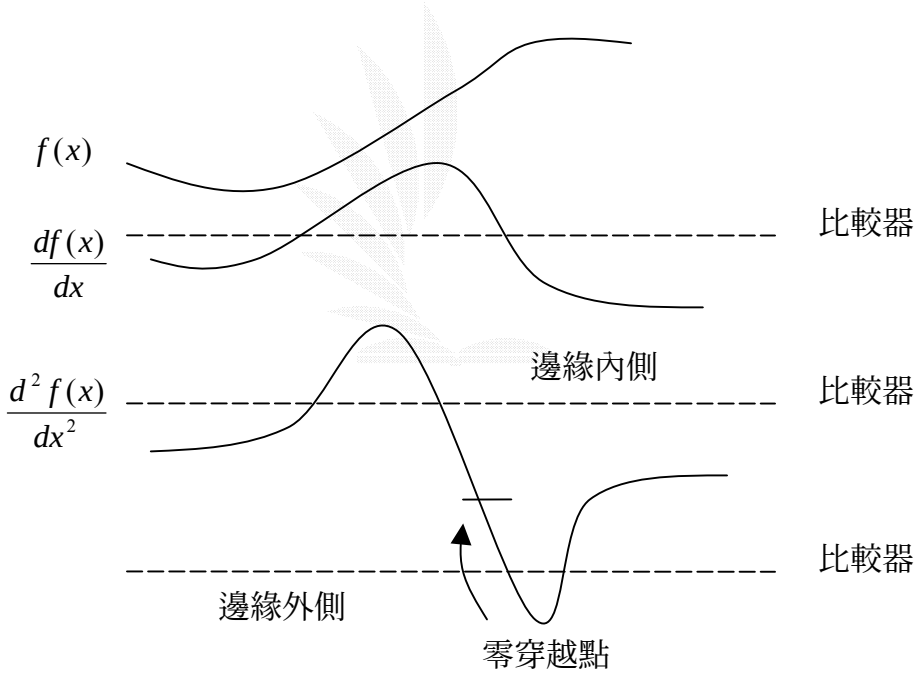


圖 三-8-2 信號一次微分和二次微分的結果

影像二次微分常用的運算式微拉普拉氏（Laplacian）運算，定義如下：

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

它為向量 $(\frac{\partial^2 f}{\partial x^2}, \frac{\partial^2 f}{\partial y^2})$ 和 $(1,1)$ 內積的結果，下面為一些常用的拉普拉氏運算

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

經由拉普拉氏運算，影像中斜率為常數（一次變化）的區域其輸出為零，舉例如下，左邊為 45°遞增的影像，右邊為拉普拉氏運算的結果，我們只計算由方形圈住的兩點輸出。

1	1	2	3	4	5	...	*x*	*x*	*x*	*x*	*x*
1	2	3	4	5			*x*	0	*x*	*x*	
2	3	4	5	6			*x*	*x*	0	*x*	
3	4	5	6				*x*	*x*	*x*	*x*	
4	5	6					*x*				
5							*x*				

三-9　臨限法

(1)介紹

影像區分：

區分乃是將構成影像的元件從背景中區分出來，如線上生產之零件。其原理在於不同物件其本身之特性各異，如書本中之文字是黑色，而背景是白色。物體和背景之不連續性，如在輸送帶上之零件，由皮帶區域進入零件區域，在邊緣必有明顯之灰度變化。利用物體本身特性將物體和背景分離較簡單，但易受陰影之影響，且物體之點數較多，而利用物體及背景之不相關性要將物體分離，須先將邊緣強化再把邊緣分離出，易受雜訊干擾，且代表物體邊緣點數較少。

臨限法：

其方法乃是將處理後的影像如雜訊濾除，邊緣強化等，之後再和某一灰度直或某灰度範圍值比較，然後將影像中之物體及背景區分出來，如圖三-9-1 為影像 $f(x,y)$ 之表示圖，定義臨限值為 T，若 $f(x,y)>T$ 的像素表物體，$f(x,y)<T$ 的像素表背景。圖三-9-2 為影像 $f(x,y)$ 另外之表示圖，其中 T_1，T_2 為臨現值，$T_1 \le f(x,y) \le T_2$ 之像點為物體，其它為背景。臨限法之關鍵在找出 T 值，使得區分誤差愈小愈佳。

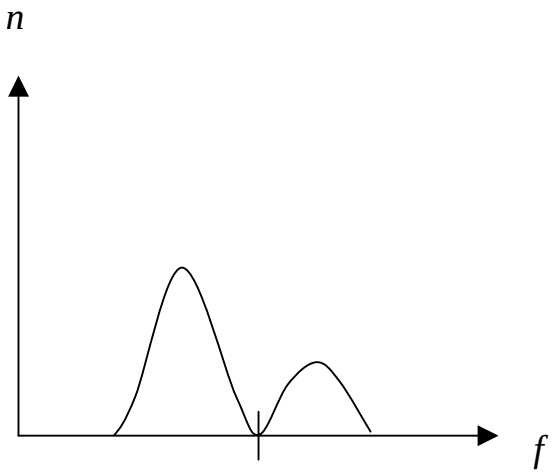


圖 三-9-1 雙峰長條圖

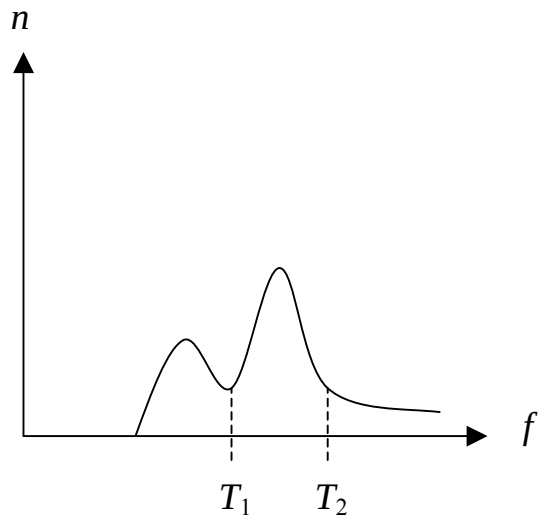


圖 三-9-2 多峰長條圖

影像之長條圖如圖三-9-3 所示。其中 f_1 表文字，f_2 表背景，但是紙張平面凹凸不一致，反射係數也不相同，並以高斯隨機分布，其實際影像長條圖如圖三-9-4 所示，離 f_1 及 f_2 之灰點值其點數愈少〈機率愈小〉，故無論選擇哪個臨限值 T，均會造成區分之誤差，及背景和文字兩者互相混淆，若加上不均勻之入射光線，則文字和背景無法清楚被分辨出。如圖 3-9.5 所示，以上的三圖可以影像的成像來表示，若 $r(x,y)$ 為反射函數，$i(x,y)$ 為照射函數，則影像 $f(x,y)$ 可寫成

$$f(x,y) = i(x,y) * r(x,y) \qquad \text{式 三-9-1}$$

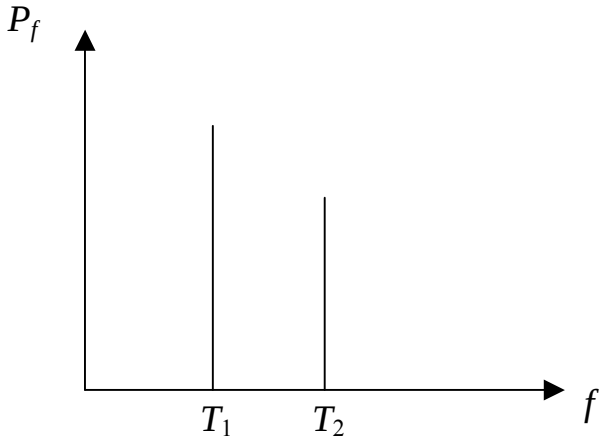


圖 三-9-3

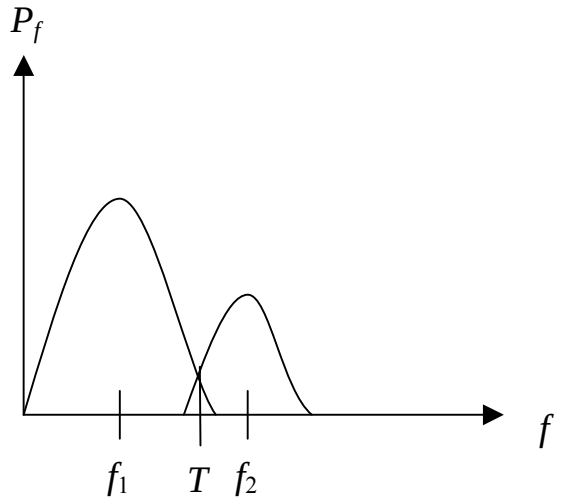


圖 三-9-4

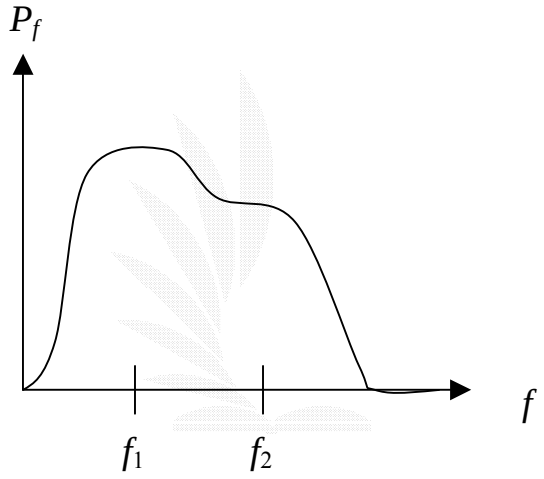


圖 三-9-5

從式三-9-1 知，若 $i(x,y)=0$ 或 $v(x,y)=0$〈無反射〉，則 $f(x,y)=0$。將式三 3-9-1 取對數

$$f'(x,y)\underline{\Delta}\log f(x,y)=\log i(x,y)+\log r(x,y)=i'(x,y)*r'(x,y)$$

假設 i 和 r 之分佈為不相關之兩機率函數，取對數只會對灰度有影響，其分佈比例不會改變，在沒有陰影（平均照明）下，$i'(x,y)$ 之機率函數 p_i' 為一個脈衝，$i'(x,y)$ 之機率函數 p_i' 為高斯分佈，則

$f'(x,y)$ 之機率函數 p_f' 為 p_i' 和 p_r' 之迴旋運算，其分佈如圖 3-8.6 所示。若照度不均勻，則運算結果如圖三-9-7 所示。

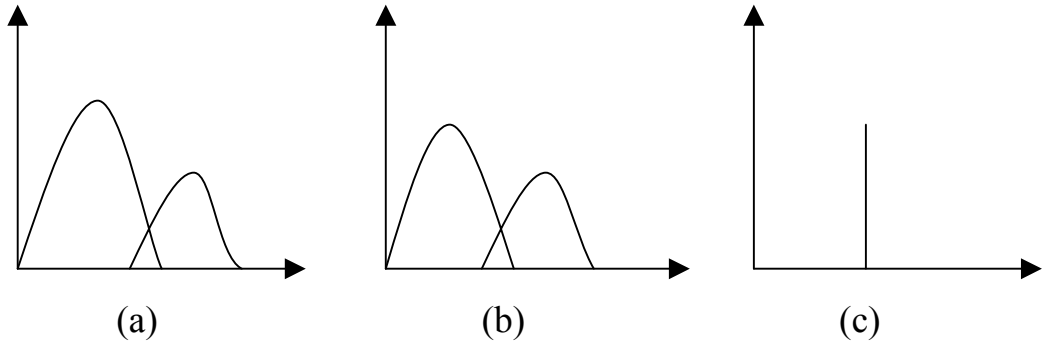


圖 三-9-6 照度均勻影像之長條圖

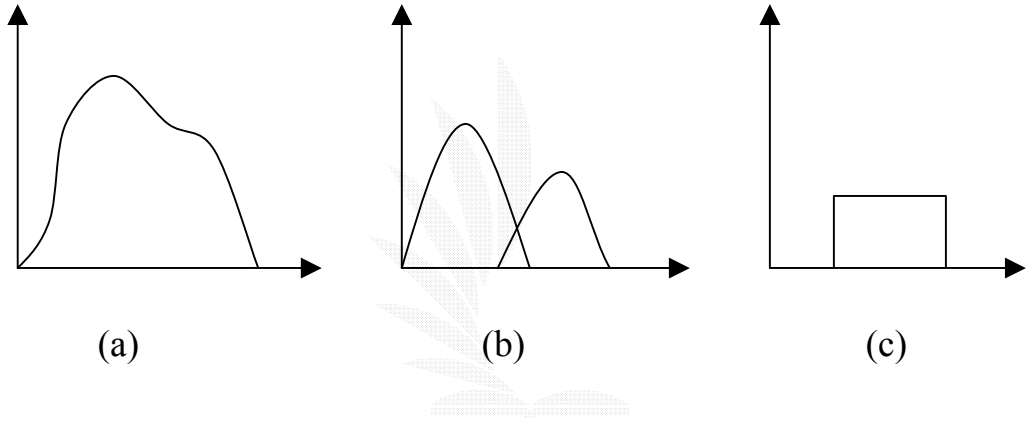


圖 三-9-7 照度不均勻影像之長條圖

圖三-9-8 為理想照度之影像區分結果，從圖中可找到臨限值 T，以影像中之物體和背景分離，區分後之影像 $g(x,y)$ 可以下式求得

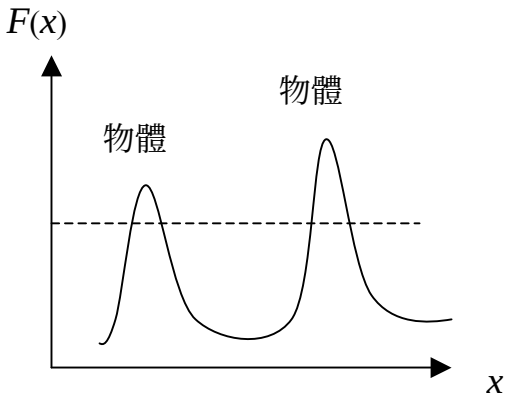


圖 三-9-8 理想照度區分

$$g(x,y)=\begin{cases}1 & if \quad f(x,y)>T\\ 0 & if \quad f(x,y)\le T\end{cases}$$

若照度不均，則無法找到 T 值以區分物體和背景，如圖三-9-9 所示，無論所選之值為何，均會造成很大的誤差。圖中之影像照度由左向右漸增，欲克服此一缺點有三種方法：1.局部比較法 2.邊緣強化 3.照明補償。一般是輸入影像 $i(x,y)$，它為照明函數，而後輸入之影像和 $i(x,y)$ 相除，將照明影響消除。

$$f'(x,y)=\frac{f(x,y)}{i(x,y)}$$

式中，$f'(x,y)$ 為照明補償後的影像，在低照度 $i(x,y)$ 小之情況下，$f'(x,y)$ 被放大，在高照度區域，$f'(x,y)$ 被縮小。若照明函數無法預先取得，可用局部比較法，臨限值 T 為 (x,y) 之函數。

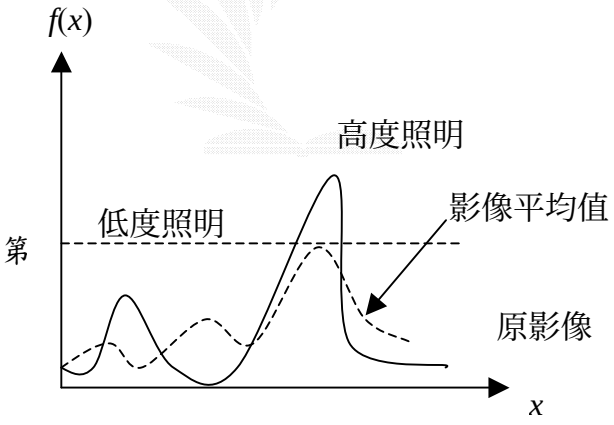


圖 三-9-9 照度不均影像區分

以

$$T(x,y)=\frac{1}{N}\sum_{w} f(x,y)$$

它是以局部平均值為局部臨限值，可用下式將影像區分。

$$g(x,y)=\begin{cases}1 & if \quad f(x,y)-T(x,y)>0\\ 0 & if \quad f(x,y)-T(x,y)\le 0\end{cases}$$

如圖三-9-9 所示，虛線部份為實線之局部平均值，兩者比較，可將物體和背景分離。而圖三-9-10 為圖三-9-9 之實線一次微分結果，從圖中展現出，照度之變化率比物體和背景之間的變化要小，取適當的臨限值，可將影像中物體之邊緣分離出來。

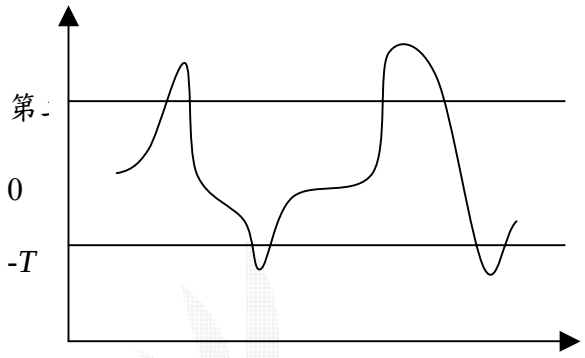


圖 三-9-10 利用微分將物體分離

(2)功用

將影像中之物體及背景區分出來

第四章　影像中的雜訊

常見的影像處理雜訊可分為：

1. 加性雜訊

加性雜訊不受影像訊號強弱影響，它出現在傳遞過程中引進的 ”通道雜訊” ，電視攝影的掃描雜訊等。此雜訊 g 可視為理想無雜訊 f 和雜訊 n 的和，即 $g = f + n$ 。

2. 乘性雜訊

乘性雜訊和影像訊號相關，它隨影像訊號而改變，如電視掃描光柵，軟片顆粒雜訊。

3. 量化雜訊：

量化雜訊之大小為原始影像和數會影像的差異，減少此雜訊方法是依灰度級機率密度函數選取最佳量化。

4. 鹽和胡椒雜訊：

即黑影像的白點，白影像的黑點。另外在轉換和反轉換之間由於誤差而引起的雜訊。

第五章　影像處理程式碼

```
/*變數*/
pixels = new int [ imgw * imgh ];
sourcePixels = new int [ imgh ][ imgw ];
PixelGrabber pg = new PixelGrabber
                    ( handling_img, 0, 0, imgw, imgh, pixels, 0, imgw);
try {
        pg.grabPixels();
    }
catch ( InterruptedException ie ){   }
int bigm = 128, lo, lomiss, q, q1, q2, lo1, lo2;
int mu = 0, twomu = 0, mu1 = 0, twomu1 = 0, mu2 = 0, twomu2 = 0;
int finalpoint = 0, changearray = 0, finalpoint1 = 0, changearray1 = 0,
    finalpoint2 = 0, changearray2 = 0;
/*
    基本上是使用以上的程式碼為主，來讀取 3*3 的 9 個點，以利用空
    間域的方法來處理之後的影像。
*/
for ( i = 0, index = 0; i < imgh; i++ )
{
    for( j = 0; j < imgw; j++ )
    {
        sourcePixels[i][j] = pixels[index++];
    }
}
for ( index = 0; index < imgw; index++ )
{
    sourcePixels[0][index] = bgColor;
}
for ( h = 0, index = 0; h < imgh; h++ )
{
    for ( w = 0; w < imgw; w++ )
    {
        if ( h == 0 || w == 0 )
        {
            pixels[index++] = sourcePixels[h][w];
        }
```

```
else if ( h == ( imgh – 1 ) || w == ( imgw – 1 ) )
{
      pixels[index++] = sourcePixels[h][w];
}
else
{
      //up
      c1 = sourcePixels[h-1][w];
      r1 = (c1&0xff0000) >> 16;
      g1 = (c1&0xff00) >> 8;
      b1 = c1&0xff;
      grayu = ( r1*3 + g1*4 + b1*2 )/9;
      //down
      c2 = sourcePixels[h+1][w];
      r2 = (c2&0xff0000) >> 16;
      g2 = (c2&0xff00) >> 8;
      b2 = c2&0xff;
      grayd = ( r2*3 + g2*4 + b2*2 )/9;
      //left
      c3 = sourcePixels[h][w-1];
      r3 = (c3&0xff0000) >> 16;
      g3 = (c3&0xff00) >> 8;
      b3 = c3&0xff;
      grayl = ( r3*3 + g3*4 + b3*2 )/9;
      //right
      c4 = sourcePixels[h][w+1];
      r4 = (c4&0xff0000) >> 16;
      g4 = (c4&0xff00) >> 8;
      b4 = c4&0xff;
      grayr = ( r4*3 + g4*4 + b4*2 )/9;
      //left down
      c5 = sourcePixels[h+1][w-1];
      r5 = (c5&0xff0000) >> 16;
      g5 = (c5&0xff00) >> 8;
      b5 = c5&0xff;
      grayld = ( r5*3 + g5*4 + b5*2 )/9;
      //right down
      c6 = sourcePixels[h+1][w+1];
      r6 = (c6&0xff0000) >> 16;
```

```
                g6 = (c6&0xff00) >> 8;
                b6 = c6&0xff;
                grayrd = ( r6*3 + g6*4 + b6*2 )/9;
                //left up
                c7 = sourcePixels[h-1][w-1];
                r7 = (c7&0xff0000) >> 16;
                g7 = (c7&0xff00) >> 8;
                b7 = c7&0xff;
                graylu = ( r7*3 + g7*4 + b7*2 )/9;
                //right up
                c8 = sourcePixels[h-1][w+1];
                r8 = (c8&0xff0000) >> 16;
                g8 = (c8&0xff00) >> 8;
                b8 = c8&0xff;
                grayru = ( r8*3 + g8*4 + b8*2 )/9;
                //read itself
                c = sourcePixels[h][w];
                r = (c&0xff0000) >> 16;
                g = (c&0xff00) >> 8;
                b = c&0xff;
                gray = ( r*3 + g*4 + b*2 )/9;
            }
        }
    }
    /*
        自動化去黑雜訊、自動化去白雜訊，乃依 smoothing filter 來參考做成。因為若完全使用 smoothing filter 會變成影像全面模糊化，因此才使用此方法來模糊白點或黑點雜訊。
    */
    // 自動化去黑雜訊
    if ( gray < 15 )   // adjust
    {
        r = ( r1 + r2 + r3 + r4 + r5 + r6 + r7 + r8 )/8;
        g = ( g1 + g2 + g3 + g4 + g5 + g6 + g7 + g8 )/8;
        b = ( b1 + b2 + b3 + b4 + b5 + b6 + b7 + b8 )/8;
        pixels[index++] = (c&0xff000000) + (r<<16) + (g<<8) + b;
    }
```

```
else
{
    pixels[index++] = (c&0xff000000) + (r<<16) + (g<<8) + b;
}
// 自動化去白雜訊
if ( gray > 235 )   // adjust
{
    r = ( r1 + r2 + r3 + r4 + r5 + r6 + r7 + r8 )/8;
    g = ( g1 + g2 + g3 + g4 +g5 + g6 + g7 + g8 )/8;
    b = ( b1 + b2 + b3 + b4 + b5 + b6 + b7 + b8 )/8;
    pixels[index++] = (c&0xff000000) + (r<<16) + (g<<8) + b;
}
else
{
    pixels[index++] = (c&0xff000000) + (r<<16) + (g<<8) + b;
}
/*
    Local mean variance enhancement 的一種應用用來減去雜訊強度。其作法為 g ( x , y ) = f ( x , y ) + y ( x ,y ) ∴ f ( x , y ) = g ( x , y )- y ( x , y )
*/
// 去黑白式雜訊
mu = ( r + r1 + r2 + r3 + r4 + r5 + r6 + r7 + r8 )/9;
twomu = ( r*r + r1*r1 + r2*r2 + r3*r3 + r4*r4 + r5*r5 + r6*r6 + r7*r7 + r8*r8 )/9;
mu1 = ( g + g1 + g2 + g3 + g4 + g5 + g6 + g7 + g8 )/9;
twomu1 = ( g*g + g1*g1 + g2*g2 + g3*g3 + g4*g4 + g5*g5 + g6*g6 + g7*g7 +
        g8*g8 )/9;
mu2 = ( b + b1 + b2 + b3 + b4 + b5 + b6 + b7 + b8 )/9;
twomu2 = ( b*b + b1*b1 + b2*b2 + b3*b3 + b4*b4 + b5*b5 + b6*b6 + b7*b7 +
        b8*b8)/9;
lo = (int) Math.sqrt ( twomu - mu*mu);
lo1 = (int) Math.sqrt ( twomu1 - mu1*mu1);
lo2 = (int) Math.sqrt ( twomu2 - mu2*mu2);
lomiss = (int) Math.sqrt( bigm );
if ( lo == 0 )
{
    finalpoint = r;
}
```

```
else
{
      changearray = (mu / lo)/2;
}
if ( lo1 == 0 )
{
      finalpoint1 = g;
}
else
{
      changearray1 = ( mu1 / lo1 ) / 2;
}
if ( lo2 == 0 )
{
      finalpoint2 = b;
}
else
{
      changearray2 = ( mu2 / lo2 ) / 2;
}
finalpoint = ( ( ( changearray ) * ( r – mu ) ) + mu ) * (1-1/lomiss);
finalpoint1 = (((changearray1)*(g-mu1))+mu1)*(1-1/lomiss);
finalpoint2 = (((changearray2)*(b-mu2))+mu2)*(1-1/lomiss);
if ( finalpoint < 0 )
{
      finalpoint = 0 - finalpoint;
}
if ( finalpoint1 < 0 )
{
      finalpoint1 = 0 - finalpoint1;
}
if (finalpoint2 < 0 )
{
      finalpoint2 = 0 - finalpoint2;
}
pixels[index++] = (c&0xff000000) + (finalpoint << 16) + (finalpoint1<<8)
                        +finalpoint2;

/*
```

$(2x - x^2)$ and $x = \frac{r}{255}$ 所以 $0 \le x \le 1$ 將$(2 - 2x)$做積分可得

*/

// 直方圖分布

```
for ( index = 0; index < imgw*imgh; index++ )
{
      c = pixels[index];
      r = (c&0xff0000) >> 16;
      g = (c&0xff00) >> 8;
      b = c&0xff;
      finalpoint = (2-r/255)*r;
      finalpoint1 = (2-g/255)*g;
      finalpoint2 = (2-b/255)*b;
      pixels[index] =
            (c&0xff000000)+(finalpoint<<16)+(finalpoint1<<8)+finalpoint2;
}
```

/*

$$m(x,y) = \frac{1}{N_w} \sum_{(x,y)\in w} f(x-i, y-1) = mu$$

$$twomu = \frac{1}{N_w} \sum_{(i,j)\in w} \left[f(x,y) - m(x,y) \right]^2$$

$$lo = \sqrt{twomu}$$

$$A(x,y) = changearray$$

$$g(x,y) = A(x,y)\left[f(x,y) - m(x,y) \right] + m(x,y)$$

*/

//Local mean variance enhancement

```
mu = (r+r1+r2+r3+r4+r5+r6+r7+r8)/9;
twomu = (r*r+r1*r1+r2*r2+r3*r3+r4*r4+r5*r5+r6*r6+r7*r7+r8*r8)/9;
mu1 = (g+g1+g2+g3+g4+g5+g6+g7+g8)/9;
twomu1 = (g*g+g1*g1+g2*g2+g3*g3+g4*g4+g5*g5+g6*g6+g7*g7+g8*g8)/9;
mu2 = (b+b1+b2+b3+b4+b5+b6+b7+b8)/9;
twomu2 = (b*b+b1*b1+b2*b2+b3*b3+b4*b4+b5*b5+b6*b6+b7*b7+b8*b8)/9;
lo = (int) Math.sqrt(twomu-mu*mu);
lo1 = (int) Math.sqrt(twomu1-mu1*mu1);
lo2 = (int) Math.sqrt(twomu2-mu2*mu2);
lomiss = (int) Math.sqrt(bigm);
changearray = (mu/lo)/2;
changearray1 = (mu1/lo1)/2;
```

```
changearray2 = (mu2/lo2)/2;
if ( lo == 0 )
{
      finalpoint = r;
}
if ( lo1 == 0 )
{
      finalpoint1 = g;
}
if ( lo2 == 0 )
{
      finalpoint2 = b;
}
finalpoint = (((changearray)*(r-mu))+mu);
finalpoint1 = (((changearray1)*(g-mu1))+mu1);
finalpoint2 = (((changearray2)*(b-mu2))+mu2);
if ( finalpoint < 0 )
{
      finalpoint = 0 - finalpoint;
}
if ( finalpoint1 < 0 )
{
      finalpoint1 = 0 - finalpoint1;
}
if ( finalpoint2 < 0 )
{
      finalpoint2 = 0 - finalpoint2;
}
pixels[index++] = (c&0xff000000)
                        + (finalpoint<<16)+(finalpoint1<<8)+finalpoint2;
/*
      和 Local mean variance enhancement 之差別在於減去最小值
*/
// Local mean variance enhancement by median
q = Math.max(Math.max(Math.max(r1,r2),
                Math.max(r3,r4)),Math.max(Math.max(r5,r6),
                Math.max(r7,r8)));
q1 = Math.max(Math.max(Math.max(g1,g2),
```

```
                Math.max(g3,g4)),Math.max(Math.max(g5,g6),
                Math.max(g7,g8)));
q2 = Math.max(Math.max(Math.max(b1,b2),
                Math.max(b3,b4)),Math.max(Math.max(b5,b6),
                Math.max(b7,b8)));
Mu = (r+r1+r2+r3+r4+r5+r6+r7+r8-q)/8;
twomu = (r*r+r1*r1+r2*r2+r3*r3+r4*r4+r5*r5+r6*r6+r7*r7+r8*r8-q*q)/8;
mu1 = (g+g1+g2+g3+g4+g5+g6+g7+g8-q1)/8;
twomu1 =
    (g*g+g1*g1+g2*g2+g3*g3+g4*g4+g5*g5+g6*g6+g7*g7+g8*g8-q1*q1)/8;
mu2 = (b+b1+b2+b3+b4+b5+b6+b7+b8-q2)/8;
twomu2 =
    (b*b+b1*b1+b2*b2+b3*b3+b4*b4+b5*b5+b6*b6+b7*b7+b8*b8-q2*q2)/8;
lo = (int) Math.sqrt(twomu-mu*mu);
lo1 = (int) Math.sqrt(twomu1-mu1*mu1);
lo2 = (int) Math.sqrt(twomu2-mu2*mu2);
lomiss= (int) Math.sqrt(bigm);
changearray = (mu/lo)/2;changearray1=(mu1/lo1)/2;changearray2=(mu2/lo2)/2;
if ( lo == 0 )
{
    finalpoint = r;
}
if ( lo1 == 0 )
{
    finalpoint1 = g;
}
if ( lo2 == 0 )
{
    finalpoint2 = b;
}
finalpoint = (((changearray)*(r-mu))+mu);
finalpoint1 = (((changearray1)*(g-mu1))+mu1);
finalpoint2 = (((changearray2)*(b-mu2))+mu2);
if ( finalpoint < 0 )
{
    finalpoint = 0 - finalpoint;
}
if ( inalpoint1 < 0 )
{
```

```
        finalpoint1 = 0 - finalpoint1;
    }
    if ( finalpoint2 < 0)
    {
        finalpoint2 = 0 - finalpoint2;
    }
    pixels[index++] = (c&0xff000000) + (finalpoint<<16) +
    (finalpoint1<<8)+finalpoint2;
    /*
        在 3×3 九點中，取其最中間值。
    */
    // median filter
    for ( i = 0; i < 9 ; i++ )
    {
        for ( j = i ; j < 9 ; j++ )
        {
            if ( data[i] > data[j] )            // 當 data[i] > data[j]時交換
            {
                temp = data[j];
                data[j] = data[i];
                data[i] = temp;
            }
        }
    }
    finalpoint = data[4];
    for ( i = 0 ; i < 9 ; i++ )
    {
        for ( j = i ; j < 9 ; j++ )
        {
            if   (data1[i] > data1[j])          // 當 data[i] > data[j]時交換
            {
                temp = data1[j];
                data1[j] = data1[i];
                data1[i] = temp;
            }
        }
    }
    finalpoint1 = data1[4];
```

```
for ( i = 0 ; i < 9 ; i++ )
{
    for ( j = I ; j < 9 ; j++)
    {
        if   (data2[i] > data2[j])            // 當 data[i] > data[j]時交換
        {
            temp = data2[j];
            data2[j] = data2[i];
            data2[i] = temp;
        }
    }
}
finalpoint2 = data2[4];
pixels[index++] =
            (c&0xff000000)+((finalpoint)<<16)+((finalpoint1)<<8)+finalpoint2;
/*
    將 3×3 中 9 點相加除以 9
*/
// smoothing filter
finalpoint = (r+r1+r2+r3+r4+r5+r6+r7+r8)/9;
finalpoint1 = (g+g1+g2+g3+g4+g5+g6+g7+g8)/9;
finalpoint2 = (b+b1+b2+b3+b4+b5+b6+b7+b8)/9;
pixels[index++] =
            (c&0xff000000)+((finalpoint)<<16)+((finalpoint1)<<8)+finalpoint2;
/*
    $\begin{bmatrix}-1 & -1 & -1\\ -1 & 8 & -1\\ -1 & -1 & -1\end{bmatrix}+\begin{bmatrix}0 & 0 & 0\\ 0 & -1 & 0\\ 0 & 0 & 0\end{bmatrix}$ 乃採用 high filter 再加上其本點值，如此會
    加強邊緣化。
*/
// high filter
much = (int)(9*r);
finalpoint = Math.min(((int)Math.abs(much-r1-r2-r3-r4-r5-r6-r7-r8)), 255);
much = (int)(9*g);
finalpoint1 = Math.min(((int)Math.abs(much-g1-g2-g3-g4-g5-g6-g7-g8)), 255);
much = (int)(9*b);
finalpoint2 = Math.min(((int)Math.abs(much-b1-b2-b3-b4-b5-b6-b7-b8)), 255);
pixels[index++] =
```

```
        (c&0xff000000)+((finalpoint)<<16) + ((finalpoint1)<<8) + finalpoint2;
/*
This mask is [ 1  -2  1 ]
             [-2   4 -2 ]
             [ 1  -2  1 ]
*/
// 拉氏運算子 1
{
    finalpoint = r*4+r5+r7+r6+r8-2*r3-2*r2-2*r1-2*r4;
    if ( 0 < finalpoint && finalpoint < 255 ){}
    else { finalpoint = r; }
    finalpoint1 = g*4+g5+g7+g6+g8-2*g3-2*g2-2*g1-2*g4;
    if ( 0 < finalpoint1 && finalpoint1 < 255 ){}
    else { finalpoint1 = g; }
    finalpoint2 = b*4+b5+b7+b6+b8-2*b3-2*b2-2*b1-2*b4;
    if ( 0 < finalpoint2 && finalpoint2 < 255){}
    else { finalpoint2 = b; }
}
/*
    This mask is [ 0  -1  0 ]
                 [-1   4 -1 ]
                 [ 0  -1  0 ]
*/
// 拉氏運算子 2
{
    finalpoint = (int)abs(r*4-r1-r2-r3-r4);
    if(0<finalpoint&&finalpoint<255){}else{finalpoint=r;}
    finalpoint1=(int)Math.abs(g*4-g1-g2-g3-g4);
    if(0<finalpoint1&&finalpoint1<255){}else{finalpoint1=g;}
    finalpoint2=(int)Math.abs(b*4-b1-b2-b3-b4);
    if(0<finalpoint2&&finalpoint2<255){}else{finalpoint2=b;}
}
/*
    This mask is ( [-1  0  +1]   [-1  -2  -1] )
                 ( [-2  0  +2] + [ 0   0   0 ] )
                 ( [-1  0  +1]   [+1  +2  +1] )
```

```
*/
// sobel
{
   finalpoint=(int)Math.sqrt((r5+2*r2+r6-r7-2*r1-r8)*(r5+2*r2+r6-r7-2*r1-r8)
                + (r7+2*r3+r5-r8-2*r4-r6)*(r7+2*r3+r5-r8-2*r4-r6) );
   if(0<finalpoint&&finalpoint<255){}else{finalpoint=r;}
   finalpoint1=(int)Math.sqrt(
   (g5+2*g2+g6-g7-2*g1-g8)*(g5+2*g2+g6-g7-2*g1-g8) +
   (g7+2*g3+g5-g8-2*g4-g6)*(g7+2*g3+g5-g8-2*g4-g6) );
   if(0<finalpoint1&&finalpoint1<255){}else{finalpoint1=g;}
   finalpoint2=(int)Math.sqrt(b5+2*b2+b6-b7-2*b1-b8)*(b5+2*b2+b6-b7-2*b1-b8)
                    + (b7+2*b3+b5-b8-2*b4-b6)*(b7+2*b3+b5-b8-2*b4-b6) );
   if(0<finalpoint2&&finalpoint2<255){}else{finalpoint2=b;}
}
/*
```

This mask is $\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} + \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$

```
*/
// Prewitt
{
   finalpoint=(int)Math.sqrt((r5+r2+r6-r7-r8-r1)*(r5+r2+r6-r7-r8-r1)
                    + (r5+r3+r7-r8-r4-r6)*(r5+r3+r7-r8-r4-r6) );
   if(0<finalpoint&&finalpoint<255){}
   else{finalpoint=r;}
   finalpoint1=(int)Math.sqrt((g5+g2+g6-g7-g8-g1)*(g5+g2+g6-g7-g8-g1) +
                    (g5+g3+g7-g8-g4-g6)*(g5+g3+g7-g8-g4-g6) );
   if(0<finalpoint1&&finalpoint1<255){}
   else{finalpoint1=g;}
   finalpoint2=(int)Math.sqrt( (b5+b2+b6-b7-b8-b1)*(b5+b2+b6-b7-b8-b1)
   + (b5+b3+b7-b8-b4-b6)*(b5+b3+b7-b8-b4-b6) );
   if(0<finalpoint2&&finalpoint2<255){}
   else{finalpoint2=b;}
}
/*
```

This mask is $\sqrt{\left(\begin{bmatrix}0&0&0\\0&1&0\\0&0&-1\end{bmatrix}^2+\begin{bmatrix}0&0&0\\1&0&-1\\0&0&0\end{bmatrix}^2\right)}$

```
*/
// Roberts
{
   finalpoint=(int)Math.sqrt((r-r6)*(r-r6)+(r4-r2)*(r4-r2));
   if(0<finalpoint&&finalpoint<255){}
   else{finalpoint=r;}
   finalpoint1=(int)Math.sqrt((g-g6)*(g-g6)+(g4-g2)*(g4-g2));
   if(0<finalpoint1&&finalpoint1<255){}
   else{finalpoint1=g;}
   finalpoint2=(int)Math.sqrt((b-b6)*(b-b6)+(b4-b2)*(b4-b2));
   if(0<finalpoint2&&finalpoint2<255){}
   else{finalpoint2=b;}
}
/*
```

This mask is $\sqrt{\begin{bmatrix}0&0&0\\-1&1&0\\0&0&0\end{bmatrix}^2+\begin{bmatrix}0&0&0\\0&1&-1\\0&0&0\end{bmatrix}^2}$

```
*/
// Roberts2
{
   finalpoint=(int)Math.sqrt((r-r2)*(r-r2)+(r-r4)*(r-r4));
   if(0<finalpoint&&finalpoint<255){}else{finalpoint=r;}
   finalpoint1=(int)Math.sqrt((g-g2)*(g-g2)+(g-g4)*(g-g4));
   if(0<finalpoint1&&finalpoint1<255){}else{finalpoint1=g;}
   finalpoint2=(int)Math.sqrt((b-b2)*(b-b2)+(b-b4)*(b-b4));
   if(0<finalpoint2&&finalpoint2<255){}else{finalpoint2=b;}
}
// 臨限法
finalpoint = r-((r+r1+r2+r3+r4+r5+r6+r7+r8)/9);
finalpoint1 = g-((g+g1+g2+g3+g4+g5+g6+g7+g8)/9);
finalpoint2 = b-((b+b1+b2+b3+b4+b5+b6+b7+b8)/9);
if ( finalpoint > 0 )
{
```

```
        finalpoint = 0;
    }
    if ( finalpoint1 > 0 )
    {
        finalpoint1=0;
    }
    if ( finalpoint2 > 0 )
    {
        finalpoint2 = 0;
    }
    if ( finalpoint <= 0 )
    {
        finalpoint = 255;
    }
    if ( finalpoint1 <= 0 )
    {
        finalpoint1 = 255;
    }
    if ( finalpoint2 <= 0 )
    {
        finalpoint2 = 255;
    }
    pixels[index++] = (c&0xff000000) + ((finalpoint)<<16) + ((finalpoint1)<<8)
                        + finalpoint2;
    /*變數*/
    pixels = new int[imgw * imgh];
    PixelGrabber pg = new
                    PixelGrabber(handling_img, 0, 0, imgw, imgh, pixels, 0, imgw);
    try {    pg.grabPixels();    }
    catch (InterruptedException ie){    }
    int g11 = 0, g21 = 0, g31 = 0, g41 = 0, g51 = 0, g61 = 0, g71 = 0, g81 = 0,
        g91 = 0, ga1=0, gb1 = 0, gc1 = 0, gd1 = 0, ge1 = 0, gf1 = 0;
    int gra =0, gra1 = 0, gra2 = 0, abc,abcd,value = 5, gray1,maxmax=0; // adjust
    value
    /*
        計算其點數 counter
    */
    for ( index = 0 ; index < imgw*imgh ; index++ )
```

```
{
    c = pixels[index];
    r = (c&0xff0000) >> 16;
    g = (c&0xff00) >> 8;
    b = c&0xff;
    gray1 = (r*3+g*4+b*2)/9;
    if ( 0 <= gray1 && gray1 < 17 )
    {
        g11++;
    }
    else if ( 17 <= gray1 && gray1 <34)
    {
        g21++;
    }
    else if ( 34 <= gray1 && gray1 < 51)
    {
        g31++;
    }
    else if ( 51 <= gray1 && gray1 < 68)
    {
        g41++;
    }
    else if ( 68 <= gray1 && gray1 < 85 )
    {
        g51++;
    }
    else if ( 85 <= gray1 && gray1 < 102)
    {
        g61++;
    }
    else if ( 102 <= gray1 && gray1 < 119)
    {
        g71++;
    }
    else if ( 119<=gray1 && gray1 < 136)
    {
        g81++;
    }
    else if ( 136 <= gray1 && gray1 < 153)
```

```
        {
            g91++;
        }
        else if ( 153 <= gray1 && gray1 < 170)
        {
            ga1++;
        }
        else if ( 170 <= gray1 && gray1 < 187)
        {
            gb1++;
        }
        else if ( 187 <= gray1 && gray1 < 204)
        {
            gc1++;
        }
        else if ( 204 <= gray1 && gray1 < 221)
        {
            gd1++;
        }
        else if ( 221 <= gray1 && gray1 < 238)
        {
            ge1++;
        }
        else
        {
            gf1++;
        }
    }
    for ( index = 0 ; index < imgw*imgh ; index++ )
    {
        c = pixels[index];
        r = (c&0xff0000)>>16;
        g = (c&0xff00)>>8;
        b = c&0xff;
        gray1 = (r*3+g*4+b*2)/9;
        if(gray1>maxmax){maxmax=gray1;}else{}
        if ( 0 <= gray1 && gray1<17)
        {
            gray = g11;
```

```
}
else if ( 17 <= gray1 && gray1 < 34)
{
      gray = g21;
}
else if ( 34 <= gray1 && gray1 < 51)
{
      gray = g31;
}
else if ( 51 <= gray1 && gray1 < 68)
{
      gray = g41;
}
else if ( 68 <= gray1 && gray1 < 85)
{
      gray = g51;
}
else if ( 85 <= gray1 && gray1 < 102 )
{
      gray = g61;
}
else if ( 102 <= gray1 && gray1 < 119)
{
      gray = g71;
}
else if ( 119 <= gray1 && gray1 < 136)
{
      gray = g81;
}
else if ( 136 <= gray1 && gray1 < 153)
{
      gray = g91;
}
else if ( 153 <= gray1 && gray1 < 170)
{
      gray = ga1;
}
else if ( 170 <= gray1 && gray1 < 187)
{
```

```
            gray = gb1;
        }
        else if ( 187 <= gray1 && gray1 <204)
        {
            gray = gc1;
        }
        else if ( 204 <= gray1 && gray1 < 221)
        {
            gray = gd1;
        }
        else if ( 221 <= gray1 && gray1 < 238)
        {
            gray = ge1;
        }
        else
        {
            gray = gf1;
        }
        abc = Math.abs(((imgw*imgh)/gray));
}
/*
```

$$S = S\min -\frac{1}{2}\ln(1-P_2(r)),\ P_r(r)=\frac{1}{abc},\ \alpha = value$$

```
*/
// 指數分佈
if ( ( ( abc – 1 ) / abc ) == 0 )
{
        gra = r;
        gra1 = g;
        gra2 = b;
}
else
{
        abcd = (int)Math.log((abc-1)/abc);
        gra = r - (1/value)*abcd;
        gra1 = g - (1/value)*abcd;
        gra2 = b - (1/value)*abcd;
}
gra = Math.min(r,255);
```

```
gra1 = Math.min(g,255);
gra2 = Math.min(b,255);
pixels[index++] = (c&0xff000000)+(gra<<16)+(gra1<<8)+gra2;
/*
```

$$S = (S\max - S\min)P_r(r) + S\min$$

$$P_r(r) = \frac{1}{abc}$$

and $S\min = 0$

```
*/
// 均勻分佈 1
if ( abc == 0 )
{
    gra = 0;
    gra1 = 0;
    gra2 = 0;
}
else
{
    gra = (maxmax/abc);
    gra1 = (maxmax /abc);
    gra2 = (maxmax /abc);
}
gra = Math.min(r,255);
gra1 = Math.min(g,255);
gra2 = Math.min(b,255);
pixels[index++] = (c&0xff000000) + (gra<<16) + (gra1<<8) + gra2;
/*
```

$S = (S\max - S\min)P_r(r) + S\min P_r(r) = \frac{1}{abc}$ and $S\min = r$ (3×3 中的點 (2,2))

```
*/
// 均勻分佈 2
if ( abc == 0 )
{
    gra = r;
    gra1 = g;
    gra2 = b;
```

```
}
else
{
      gra = (((maxmax -r)/abc)+r);
      gra1 = (((maxmax -g)/abc)+g);
      gra2 = (((maxmax -b)/abc)+b);   }
      gra = Math.min(r,255);
      gra1 = Math.min(g,255)
      gra2 = Math.min(b,255);
      pixels[index++] = (c&0xff000000) + (gra<<16) + (gra1<<8) + gra2;
/*
```

$$S = S\min\left[\frac{S\max}{S\min}\right]^{P_r(r)}$$ Smin 為 3×3 點中，點(2,2)

```
*/
// 雙曲線分佈
if ( r == 0 )
{
      gra = 0;
}
if ( g == 0 )
{
      gra1 = 0;
}
if ( b == 0 )
{
      gra2 = 0;
}
else
{
      gra = r*(int)Math.pow(maxmax /r,1/abc);
      gra1 = g*(int)Math.pow(maxmax /g,1/abc);
      gra2 = b*(int)Math.pow(maxmax /b,1/abc);
}
gra = Math.min(r,255);
gra1 = Math.min(g,255);
gra2 = Math.min(b,255);
pixels[index++] = (c&0xff000000) + (gra<<16) + (gra1<<8) + gra2;
/*
```

$$S = S\min + \left[2\alpha^2 \ln\left(\frac{1}{1 - P_r(r)}\right) \right]^{1/2}$$

*/

// 雷利分佈

```
if ( abc != 1 )
{
      if ( ( abc – 1 ) / abc ==1 )
      {
            gra = r;
            gra1 = g;
            gra2 = b;
      }
      else
      {
            abcd = ( int ) Math.log (abc/(abc-1));
            gra = r + ( int ) Math.sqrt (2*value*value*abcd);
            gra1 = g+( int ) Math.sqrt (2*value*value*abcd);
            gra2 = b+( int ) Math.sqrt (2*value*value*abcd);
      }
}
else
{
      gra = r;
      gra1 = g;
      gra2 = b;
}
gra = Math.min(r,255);
gra1 = Math.min(g,255);
gra2 = Math.min(b,255);
pixels[index++] = (c&0xff000000) + (gra<<16) + (gra1<<8) + gra2;
```

第六章　操作範例

我們的醫學影像瀏覽系統有分兩個部分，第一部分是指醫師登入和選擇所需要的影像所在，第二部分即是使用 JavaApplet 來瀏覽影像及作適當的處理。

在開啟網站後所看到的是醫師登入窗格 (圖一) ，輸入正確的醫師代碼和密碼，如果登入成功會進入選單 (圖二) ，可以選擇利用病人的名字 (Query by patient name) 或代碼 (Query by patient id)來查詢，和登出 (logout)。

我們先選擇用病人的名字來查詢，會進入查詢病人名字的畫面 ，輸入適當的名字，會列出相同姓名的病人，醫師可以利用所列出的相關資料來選擇病人，接著會進入病人相關的看診紀錄 (圖三) ，選擇所需的看診紀錄則立即進入 Applet 的部分 (圖六) 。

我們也可以用病人的代碼來查詢 (圖四)，輸入適當的代碼，會列出病人的看診紀錄 (圖四)，接下來就是點選需要觀看的紀錄進入 Applet (圖六)。

當不小心輸入不存在的病人名稱或代碼，畫面上也會有小小的提醒文字 (圖五)。

在進入 JavaApplet 後，我們先介紹基本了按鈕操作，用滑鼠雙擊要觀察的縮圖後，會顯示該張原始圖像(圖七)；用滑鼠再圖片上托曳，可以移動檢視範圍(圖八)；按[下一張]，會到編號的下一張(圖九)；按[上一張]，會到編號的上一張，按[適當大小] (圖九)；按[旋轉+]　會順時針向旋轉 90 度(圖十)；按[旋轉-] 會逆時針向旋轉 90 度(圖十一)。

介紹完基本的登入過程和基本操作後，接下來，就要介紹進階的處理過程。

第一，先介紹 image distribution 這個 Label(圖十二)

裡面按鍵的功能主要是 1.提高影像對比度，亦即增強影像。2.影像分佈的功能大致上是用來處理影像中顏色不均衡而導致影像不清楚，例如：影像過白或過黑的時侯，就可以利用此種方法將顏色依一定比例的拉開，讓影像更為清楚。在這裡我們提供了幾種方法：直分圖分佈、指數分佈、均勻分佈 1、均勻分佈 2、雙曲線分佈、雷利分佈。

image distribution – 直方圖分佈

因為此功能會將影像的分佈拉開的程度很大,所以對於很黑或很白的影像有較佳的效果，反而在正常的影像不適合此方法。(圖十三)

image distribution – 指數分佈

此功能會將影像的分佈拉開的程度不會依照比例式拉開,對於很黑或很白的影像不一定會有較佳的效果，但會將影像的分佈依指數的比例拉開，所以即使使用正常的影像，也不會失去太多。(圖十四)

image distribution – 均勻分佈 1

因為此功能會將影像的分佈拉開的程度不會依照比例式拉開,所以對於較黑的影像不會有較佳的效果，反而對較白的影像有較佳的效果。所以不適合用在一般正常的影像上。(圖十五)

image distribution – 均勻分佈 2

因為此功能會將影像的分佈拉開的程度不會依照比例式拉開,所以對於很黑或很白的影像不一定會有較佳的效果，所以即使使用正常的影像，也不會失去太多。(圖十六)

image distribution – 雙曲線分佈

因為此功能會將影像的分佈拉開的程度不會依照比例式拉開,所以對以很黑或很白的影像不一定會有較佳的效果，所以即使使用正常的影像，也不會失去太多。(圖十七)

image distribution – 雷利分佈

因為此功能會將影像的分佈拉開的程度不會依照比例式拉開,所以對以很黑或很白的影像不一定會有較佳的效果，所以即使使用正常的影像，也不會失去太多。(圖十八)

第二，介紹 image filter，至個 Label 中(圖十九)

影像濾波器的功能在於使影像更為清晰,所以在部份的功能上亦有去雜訊的效果，值得注意的是，在 x 光圖片中（黑白），使用高通濾波會使輪廓更為清晰。在超音波圖片中（彩色），若使用高通濾波則會使影像變得難以辨識。這時反而使用高通濾波，這個 tag 上，提供了對比、傳統 3D、Local mean variance enhancement、Local mean variance enhancement by median、median filter、smoothing filter、high filter

image filter – 對比

會突顯影像中色彩劇烈變化的部分，而其他部分幾乎會維持不變(圖二十)

image filter –Local mean variance enhancement

長條圖強影像是利用整體資訊來強化影像，使整張影像不偏黑或偏暗且對比最強烈，但此種方法對於局部影像偏黑或偏暗則無法得到強化的效果。(圖二十一)

image filter –Local mean variance enhancement by median

長條圖強影像是利用整體資訊來強化影像，使整張影像不偏黑或偏暗且對比最強烈，但此種方法對於局部影像偏黑或偏暗則無法得到強化的效果。如果當使用者想降低圖形變化的程度，不要像 Local mean variance enhancement 這樣劇烈，就使用這種方法。(圖二十二)

image filter – medium filter(圖二十三)

此濾波適合用在下列條件上：

1.對某些輸入訊號，中值濾波有不變性

2.中值濾波具消除雜訊功能

3.中值濾波不變的頻譜特性

注意:使用於彩色的超音波圖形時會有一定的效果

image filter – smootjing filter(圖二十四)

此種濾波器主要用來去除雜訊，去除高頻信號成份和來重新取樣。

警告：因去除高頻信號成份，去除雜訊的特性將導致影像模糊。

image filter – high filter(圖二十五)

它可以銳化影像，而銳化的主要目的是增強影像中的細微部份或增強已經模糊了的細節。模糊原因可能是影像獲取的特定方法的誤差，也可以是方法的自然效應。

此效果最適合用在醫生使用黑白 X 光圖片，使其較不清楚的地方有較好的表現。

第三，介紹 gandient filter 這個 Label(圖二十六)

一次微分對於邊緣過渡叫窄的影像效果較好，如果邊緣變化較緩，

可利用二次微分幫忙偵測邊緣。

1.邊緣檢測: 用來檢測有意義的不連續特徵的最常見方法，這是因為孤立的點和細線在大多數的實際應用中是較少出現的。

2.對比增強處理: 為了使模糊的影像變得更加清楚而採用對比增強，而影像模糊之主要原因乃是對其做平均或積分運算，因此若做反運算就使其變清晰。

警告:此 tag 會容易因為圖形種類的不同而有很大的差異若使用者在使用此 tag 中的功能時若有影像突然全部一片黑時請將圖形還原後在與以其他此 tag 中的功能再嘗試

gandient filter – 拉氏運算 1(圖二十七)

拉普拉氏（Laplaction）運算為二次微分,而本功能微常用的拉普拉氏（Laplaction）運算之一倘若影像變的一片漆黑表示圖形的影響邊緣較不明顯

注意:本功能會因影像種類的不同而有很大的變化倘若影像變的一片漆黑建議該張影像若先以用高通濾波加強圖形的影響邊緣

gandient filter – 拉氏運算 2(圖二十八)

拉普拉氏（Laplaction）運算為二次微分,而本功能微常用的拉普拉氏（Laplaction）運算之一倘若影像變的一片漆黑表示圖形的影響邊緣較不明顯

注意:本功能會因影像種類的不同而有很大的變化倘若影像變的一片漆黑建議該張影像若先以用高通濾波加強圖形的影響邊緣

gandient filter – sobel(圖二十九)

sobel 運算為一次微分的一種, 一次微分對於邊緣過渡叫窄的影像效果較好且 Sobel 濾波器是由微分運算和低通運算所結合,固有降低雜訊的效果。

gandient filter – prewitt(圖三十)

prewitt 運算為一次微分的一種, 一次微分對於邊緣過渡叫窄的影像效果較好且 prewitt 濾波器是由微分運算和低通運算所結合，固有降低雜訊的效果。

gandient filter – Roberts 1(圖三十一)

Roberts 運算為一次微分的一種, 一次微分對於邊緣過渡叫窄的影像效果較好 Roberts1 和 Roberts2 的差別在於辨別方向之不同

注意:可選用 Roberts1 和 Roberts2 的任一種即可

gandient filter – Roberts 2(圖三十二)

Roberts 運算為一次微分的一種, 一次微分對於邊緣過渡叫窄的影像效果較好 Roberts1 和 Roberts2 的差別在於辨別方向之不同

注意:可選用 Roberts1 和 Roberts2 的任一種即可

第四，介紹 image deriven(圖三十三)

影像區分：

區分乃是將構成影像的元件從背景中區分出來，如線上生產之零件。其原理在於⑴不同物件其本身之特性各異，如書本中之文字是黑色，而背景是白色。⑵物體和背景之不連續性。利用物體本身特性將物體和背景分離較簡單，但易受陰影之影響，且物體之點數較多，而利用物體及背景之不相關性要將物體分離，須先將邊緣強化再把邊緣分離出，易受雜訊干擾，且代表物體邊緣點數較少。

注意: 須先將邊緣強化再把邊緣分離出，去除雜訊干擾後再做

image deriven – 輪廓突顯(圖三十四)

用來突顯影像當中物體的輪廓,經過輪廓突顯處理後的影像會比原始影像來得暗

image deriven – 臨限法(圖三十五)

其方法乃是將處理後的影像如雜訊濾除，邊緣強化等，之後再和某一灰度直或某灰度範圍值比較，然後將影像中之物體及背景區分出來。臨限法之關鍵在找出 T 值，使得區分誤差愈小愈佳。

最後我們舉幾個範例，在使用後會有哪些效果

特別示範

1. 黑白圖形影像，如 X 光片、斷層掃描…(圖三十六)

high filter(圖三十七)

high filter to 指數分佈(圖三十八)

high filter to 雙曲線分佈(圖三十九)

high filter to 雷利分佈(圖四十)

high filter to 均勻分佈 2(圖四十一)

high filter to 均勻分佈 1(圖四十二)

high filter to 直方圖分佈(圖四十三)

high filter to 雙曲線分佈 to 臨限法(圖四十四)

high filter to 雙曲線分佈 to 輪廓突顯(圖四十五)

smoothing filter(圖四十六)

local mean value(圖四十七)

local mesn vari by median(圖四十八)

sobel(圖四十九)

prewitt(圖五十)

roberts1(圖五十一)
robert2(圖五十二)

臨限法(圖五十三)

直方圖分佈 to 均勻分佈 1(圖五十四)

直方圖分佈 to 均勻分佈 2(圖五十五)

2. 彩色圖形影像，如超音波圖…(圖五十六)

hight filter(圖五十七)

median filter(圖五十八)

smoothing filter(圖五十九)

median filter to 指數分佈(圖六十)

median filter to 雙曲線分佈(圖六十一)

median filter to 雷利分佈(圖六十二)

median filter to 均勻分佈 2(圖六十三)

median filter to 均勻分佈 1(圖六十四)

median filter to 直方圖分佈(圖六十五)

median filter to 直方圖分佈 to 臨限法(圖六十六)

median filter to 直方圖分佈 to 輪廓突顯(圖六十七)

local mean value(圖六十八)

local mesn vari by median(圖六十九)

sobel(圖七十)

prewitt(圖七十一)

roberts1(圖七十二)

roberts2(圖七十三)

臨限法(圖七十四)

輪廓突顯(圖七十五)

直方圖分佈 to roberts1(圖七十六)

直方圖分佈 to roberts2(圖七十七)

3. 特殊圖形影像，如帶有雜訊的圖…(圖七十八)

smoothing to smoothing to median to high(圖七十九)

median to median to median to 均勻分布 1 to 自動化黑雜訊(圖八十)

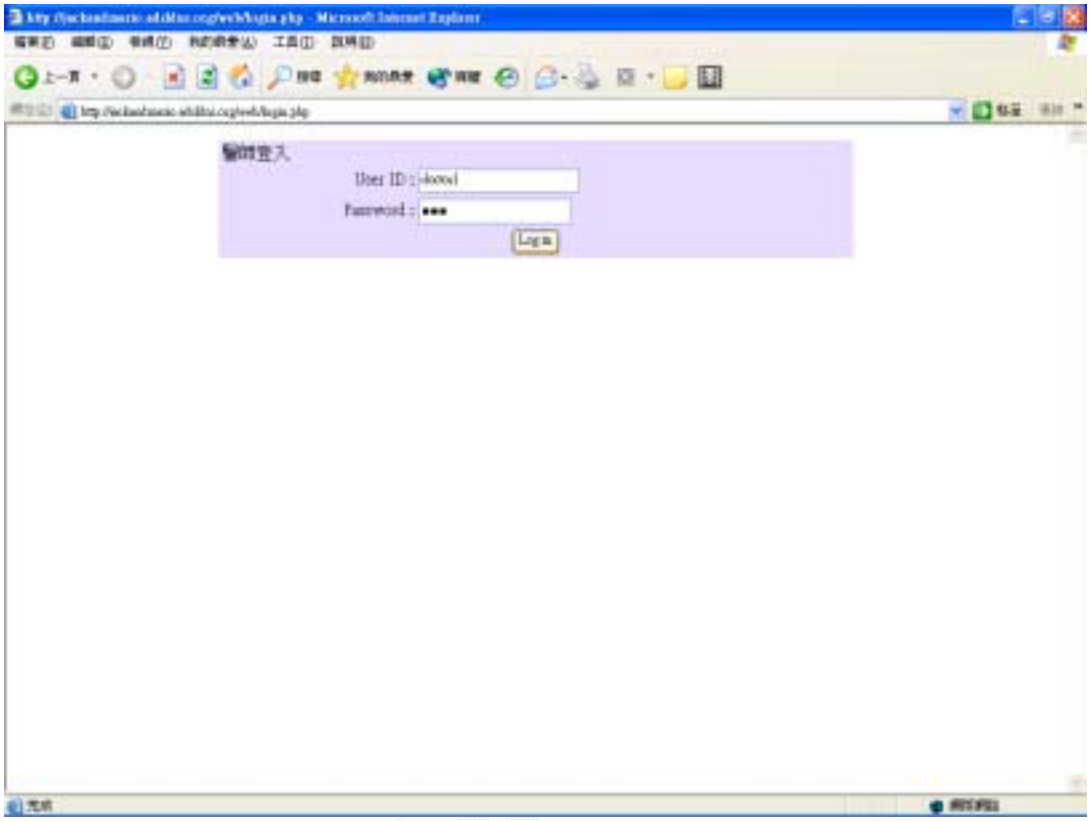
median to median to median to robert1(or robert2) (圖八十一)

median to median to smoothing to smoothing to sobel (圖八十二)

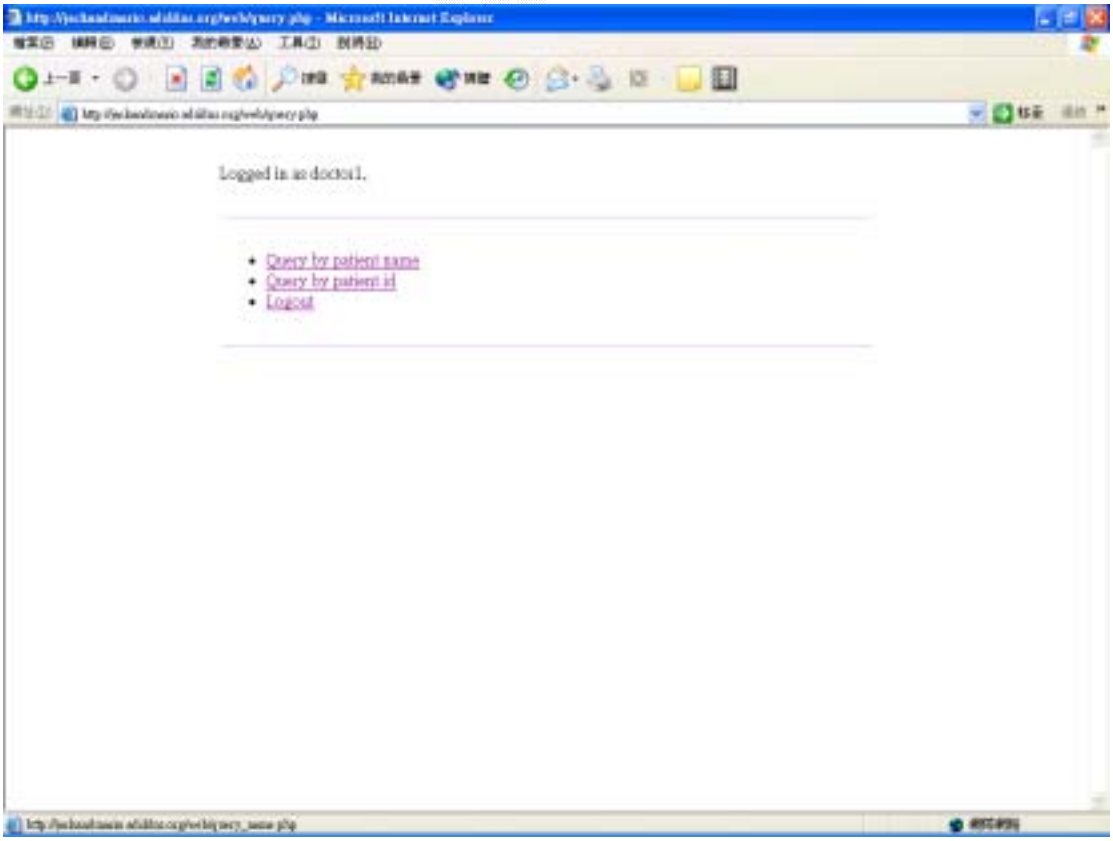
median to median to smoothing to smoothing to sobel to 臨限法
(圖八十三)

median to median median to smoothing to smoothing to prewitt(圖八十四)

圖集


User ID :
Password :
Login

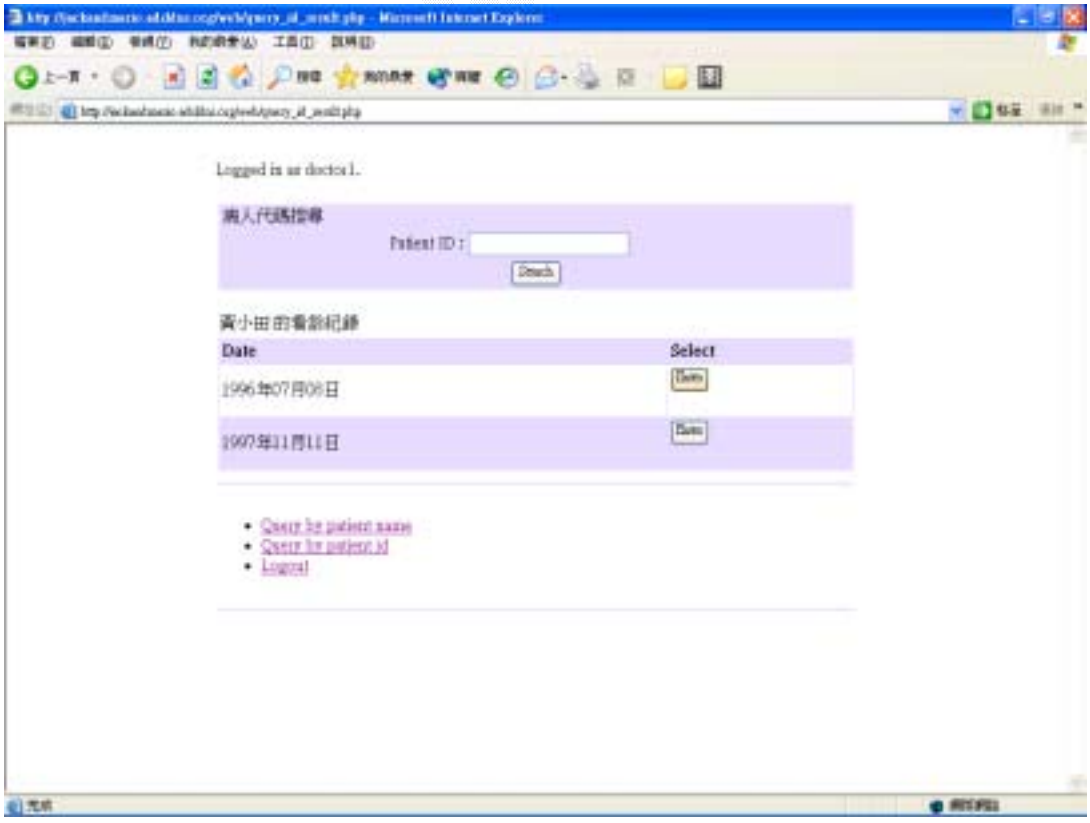
圖一


Logged in as doctor1.
Query by patient name
Query by patient id
Logout

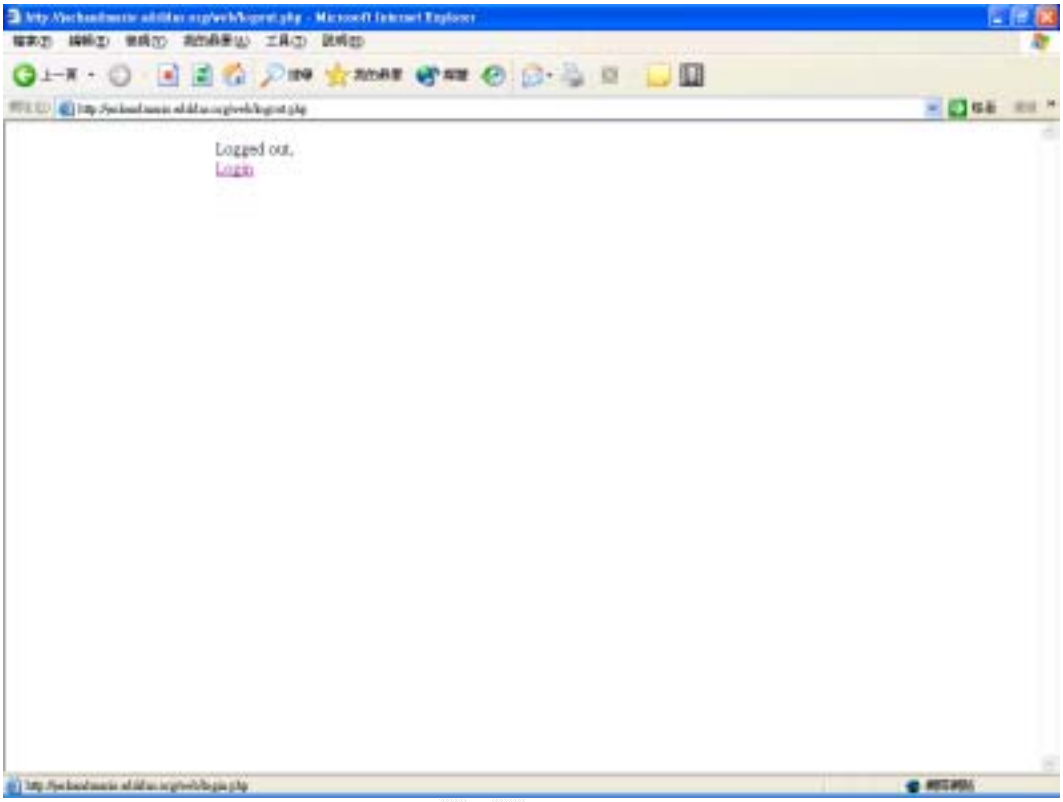
圖二


Logged in as doctor1.
病人名字搜尋
Patient Name :
Search
查詢 大頭
Patient Name
Patient ID
Telephone No.
QUERY
大頭
F5
546454
Query
大頭
F6
54080806
Query
Query by patient name
Query by patient id
Logout

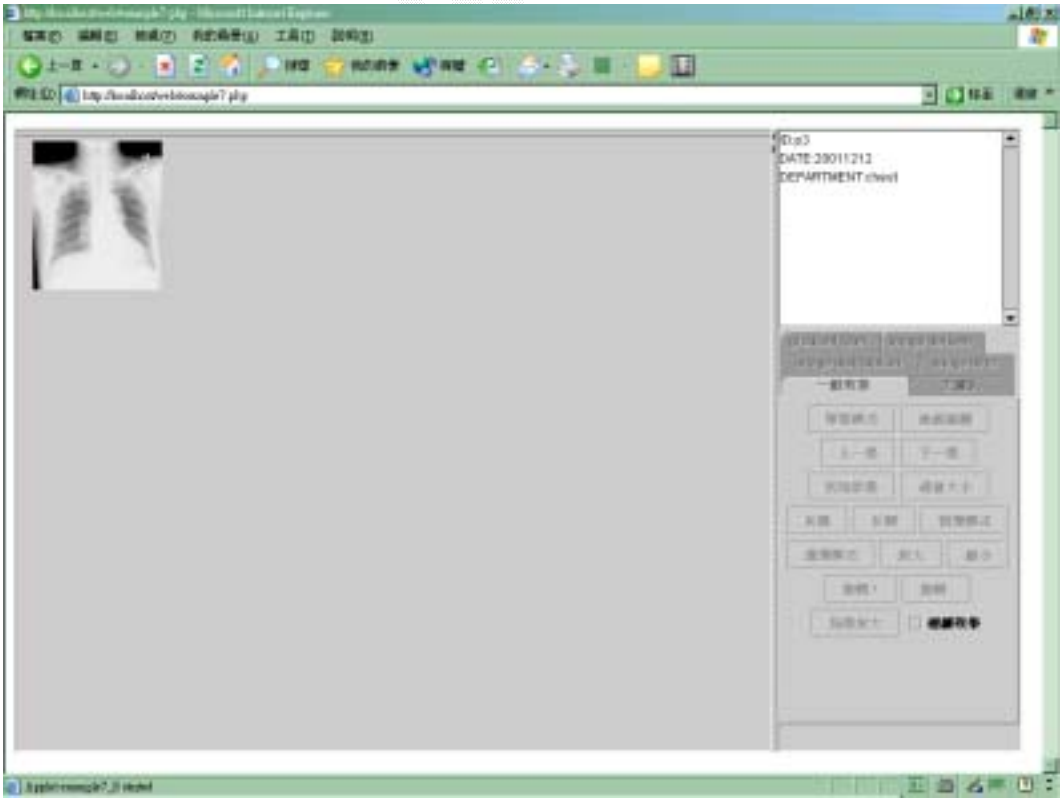
圖三


Logged in as doctor1.
病人代碼搜尋
Patient ID :
Search
黃小田的看診紀錄
Date
Select
1996年07月08日
Query
1997年11月11日
Query
Query by patient name
Query by patient id
Logout

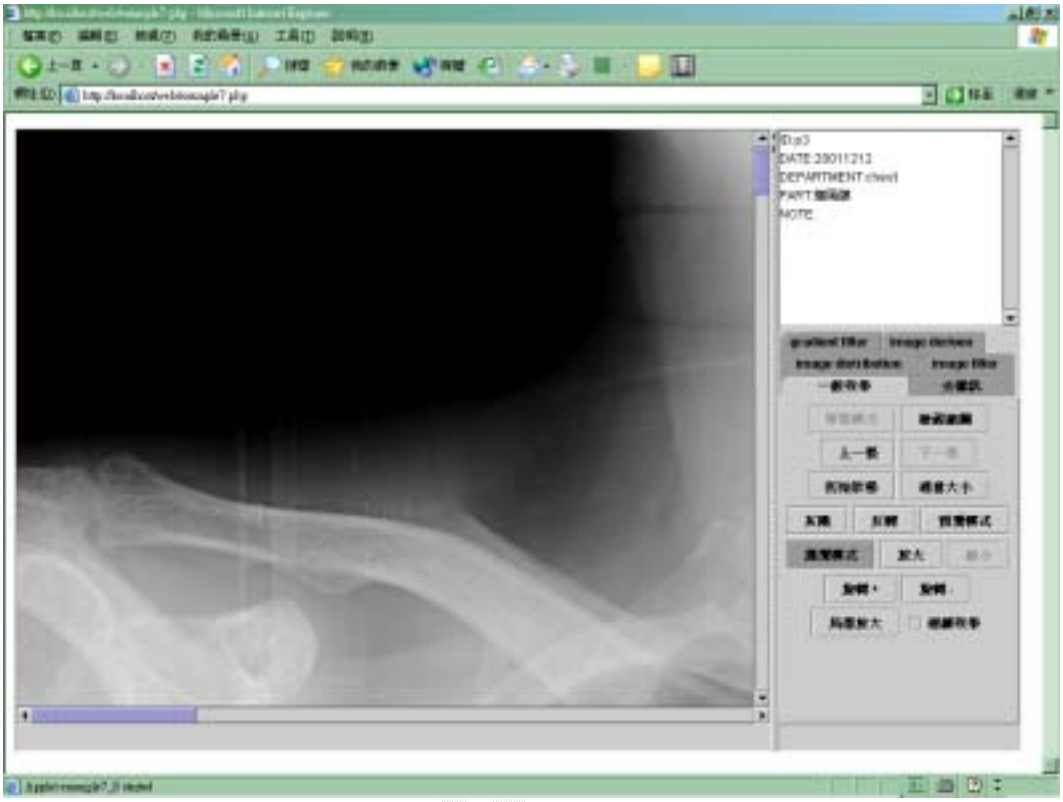
圖四



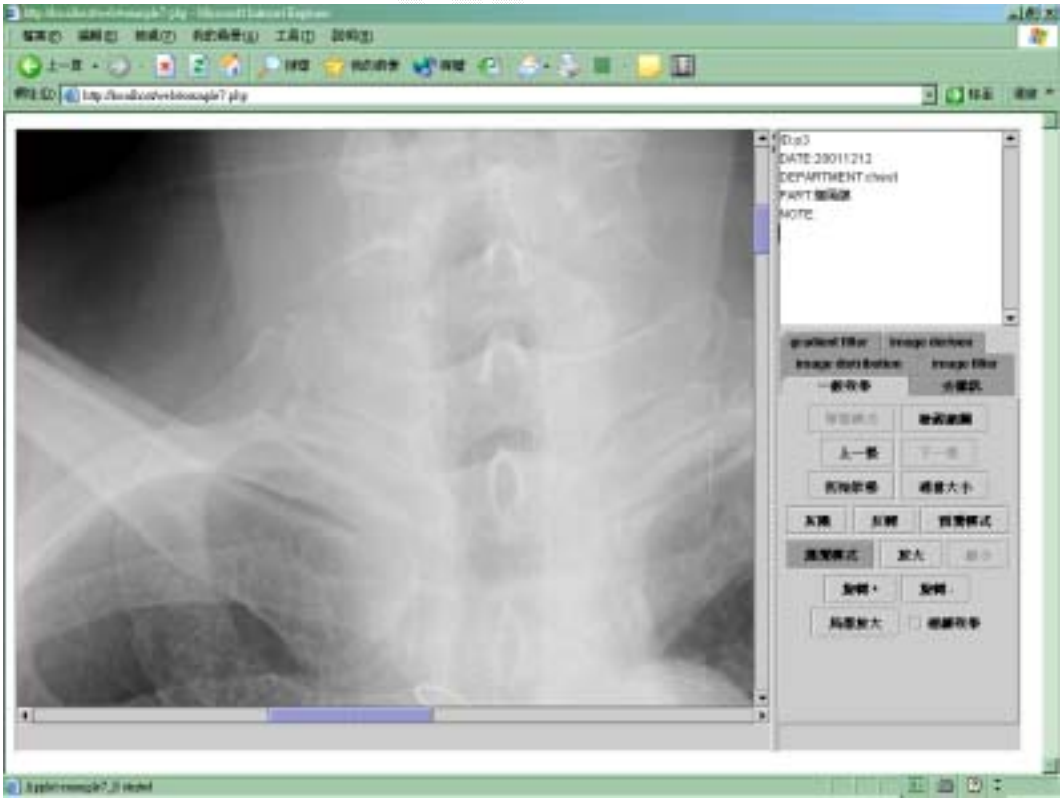
圖五



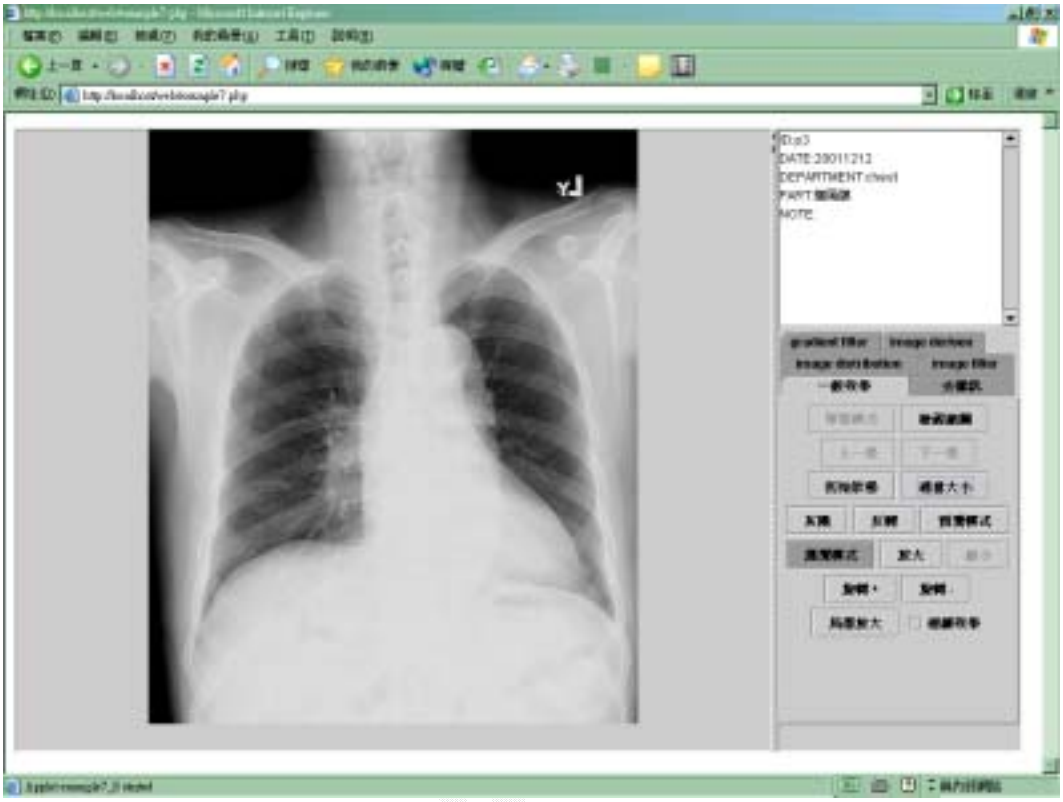
圖六



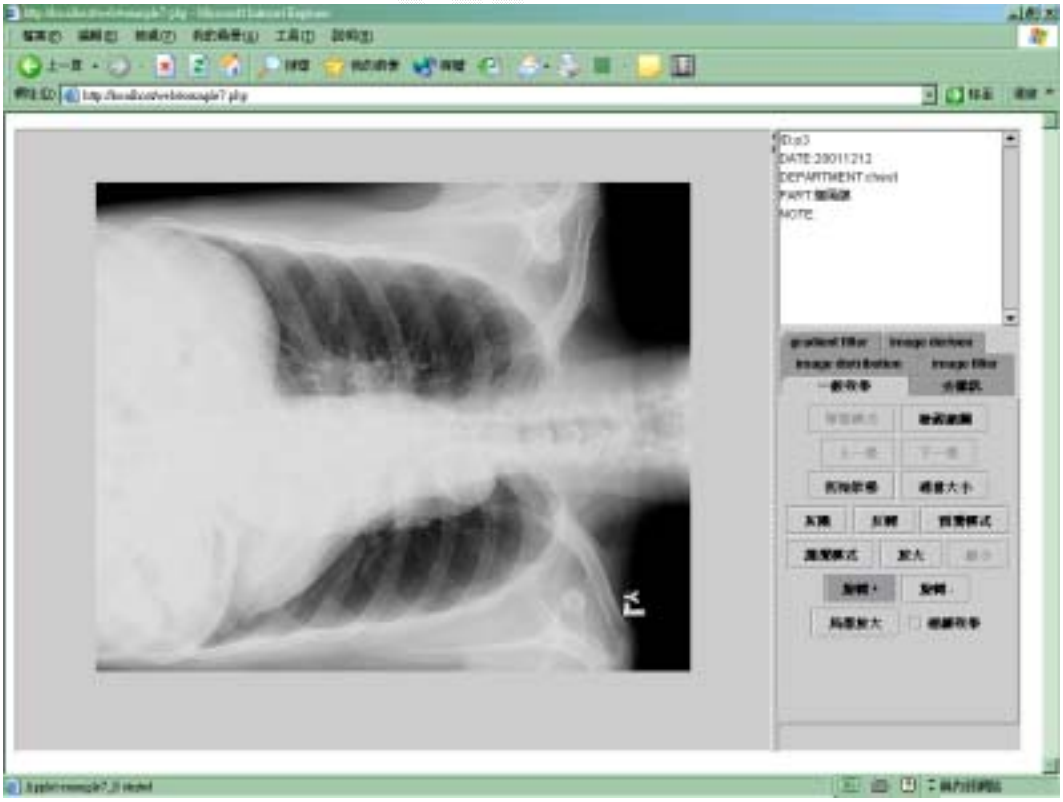
圖七



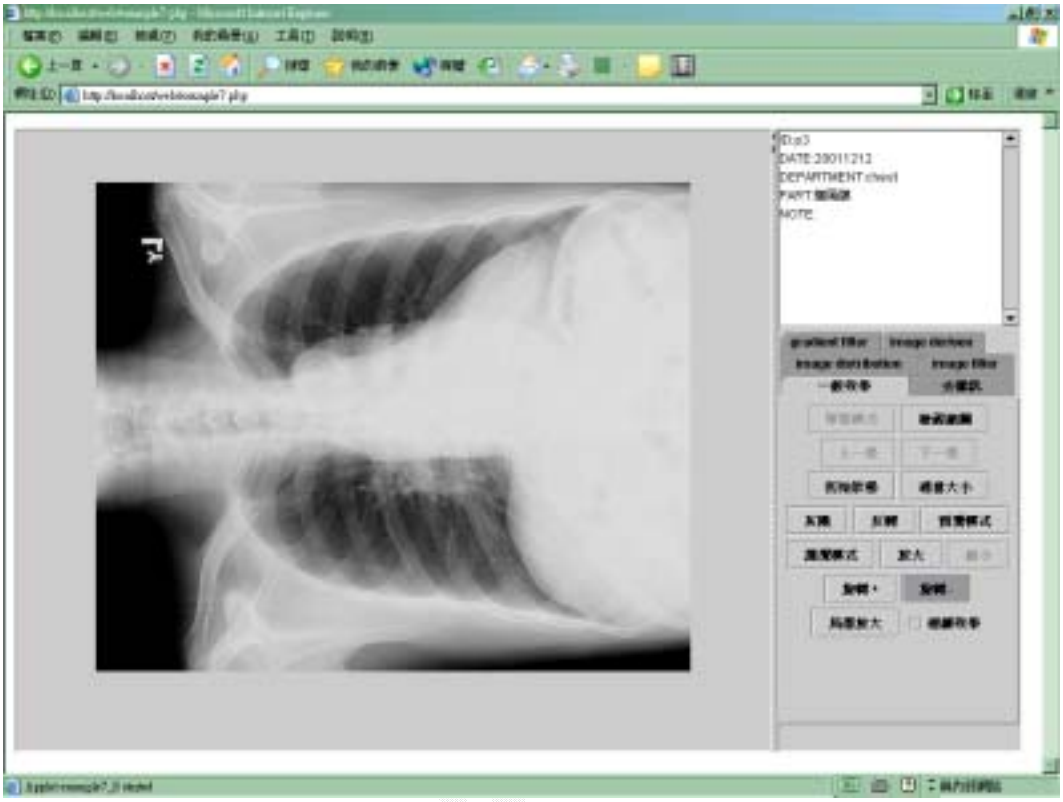
圖八



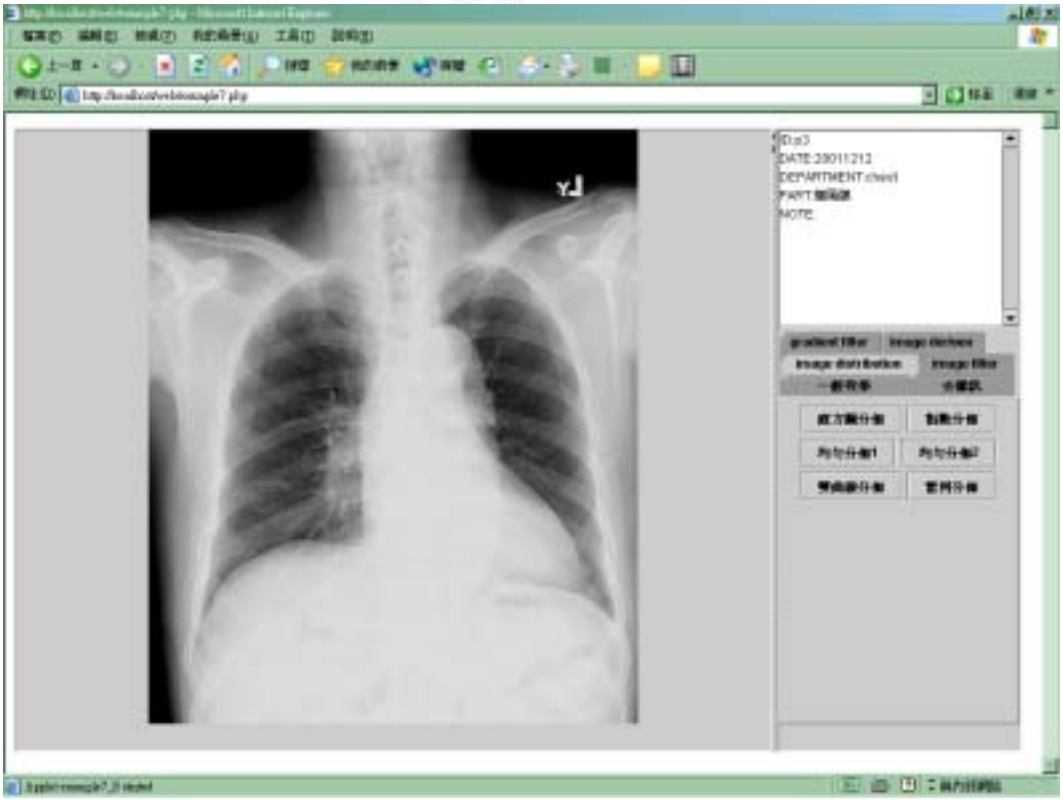
圖九



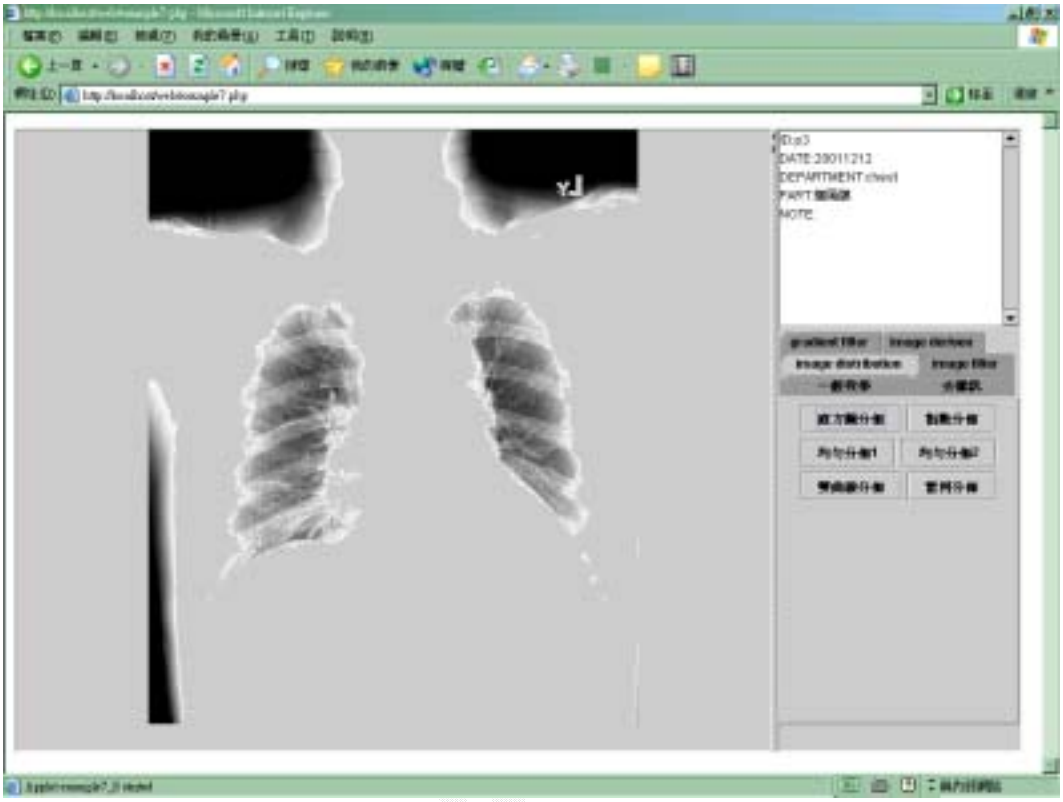
圖十



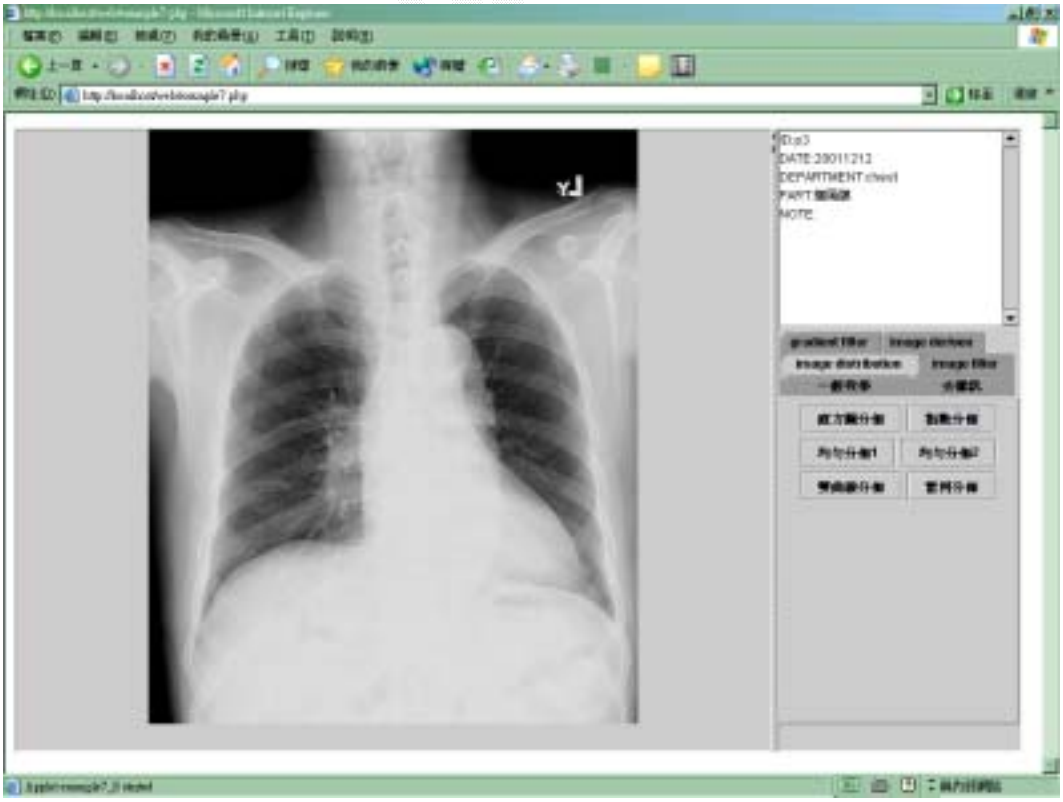
圖十一



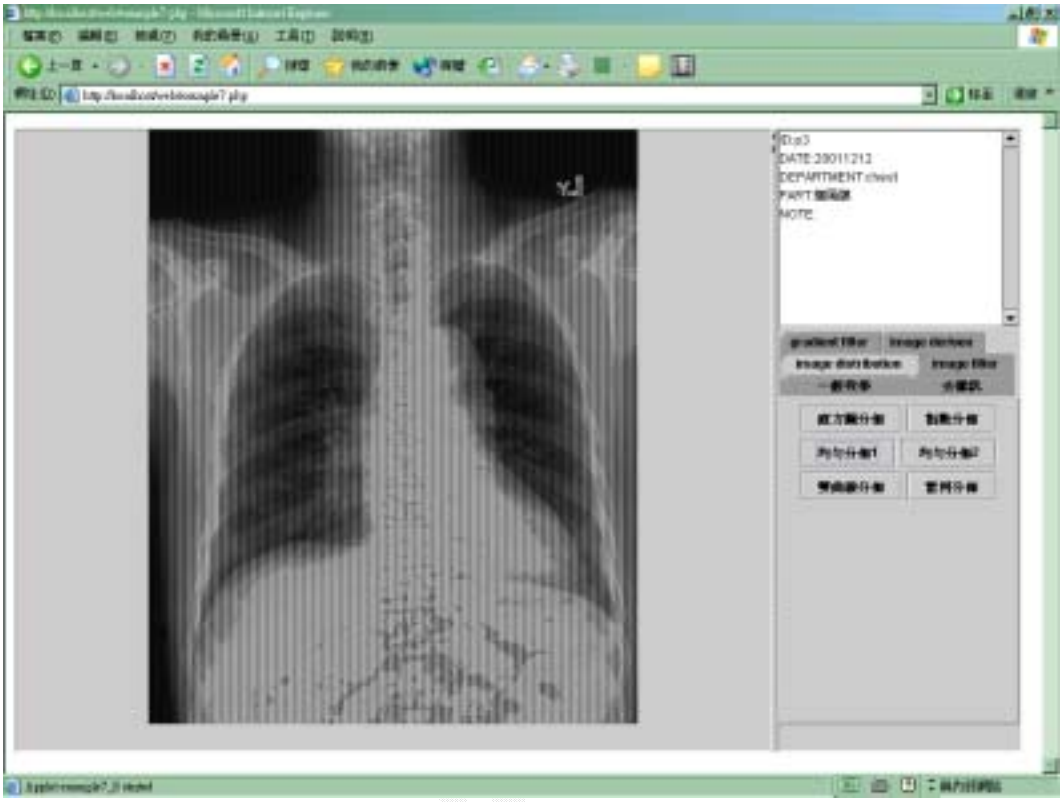
圖十二



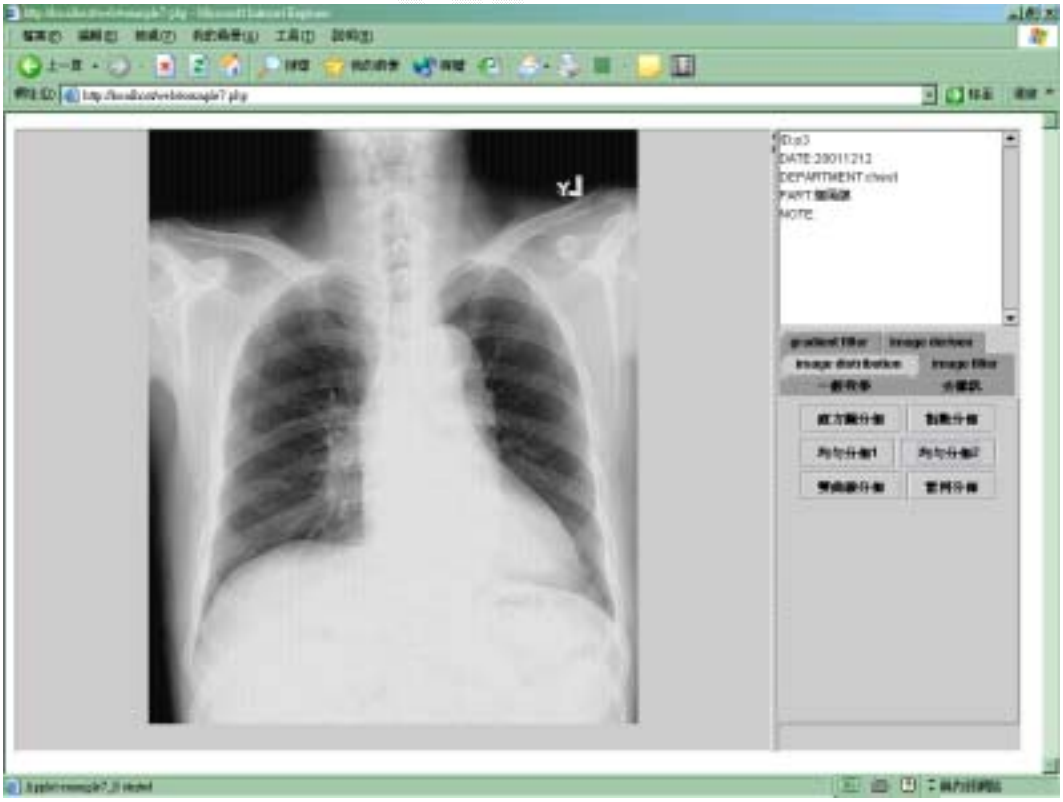
圖十三



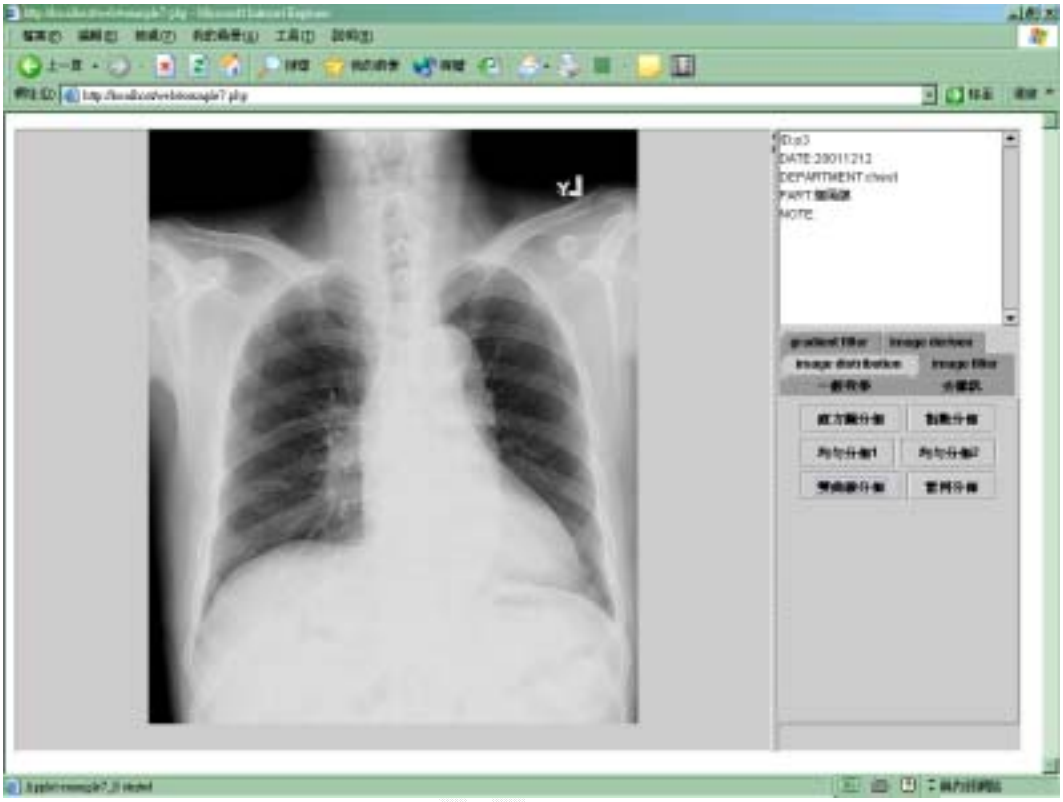
圖十四



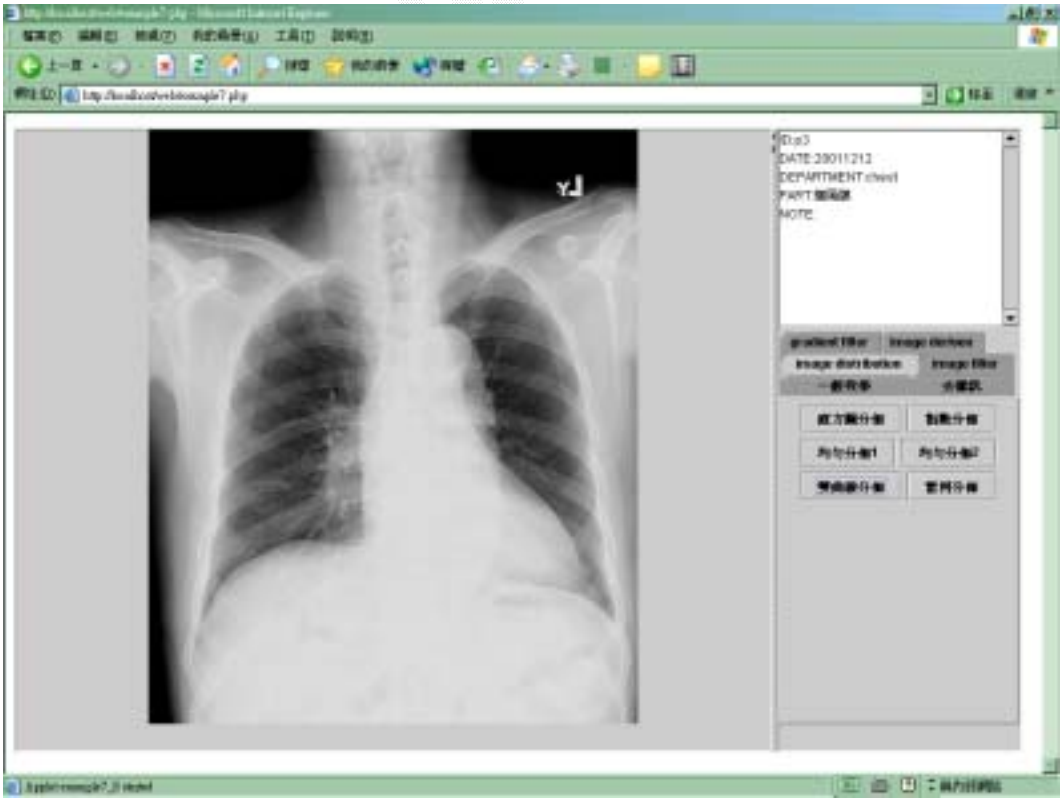
圖十五



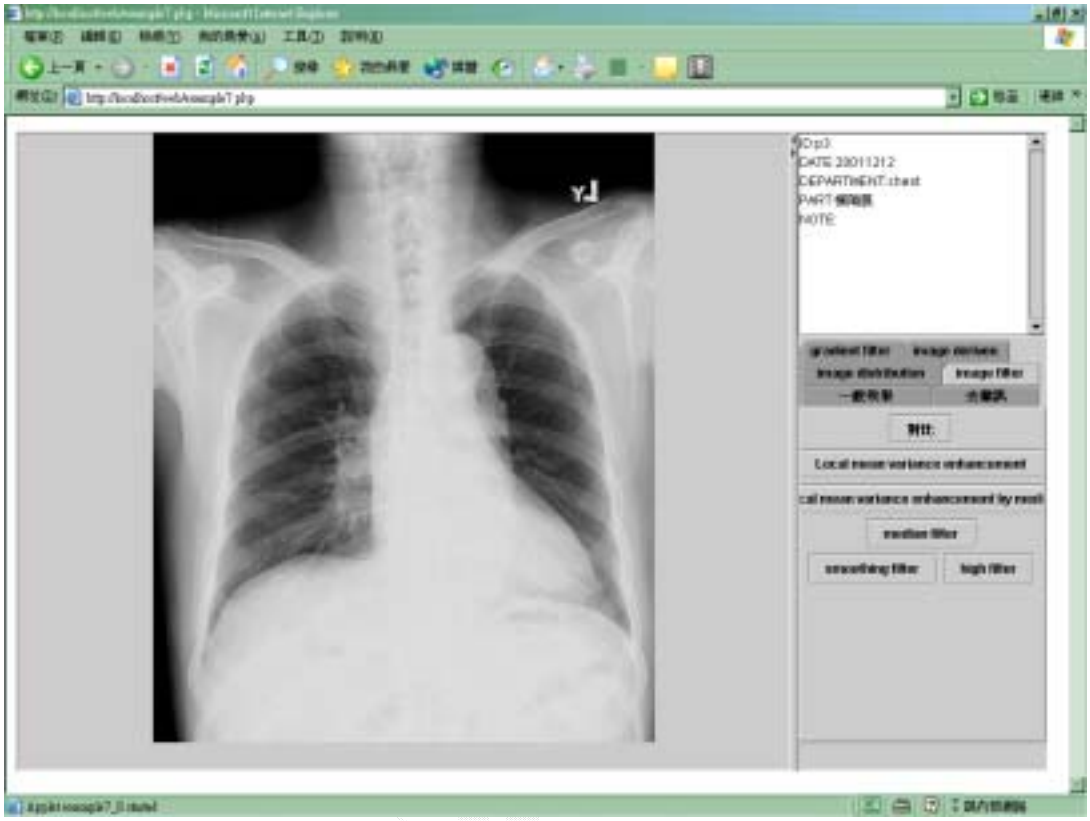
圖十六



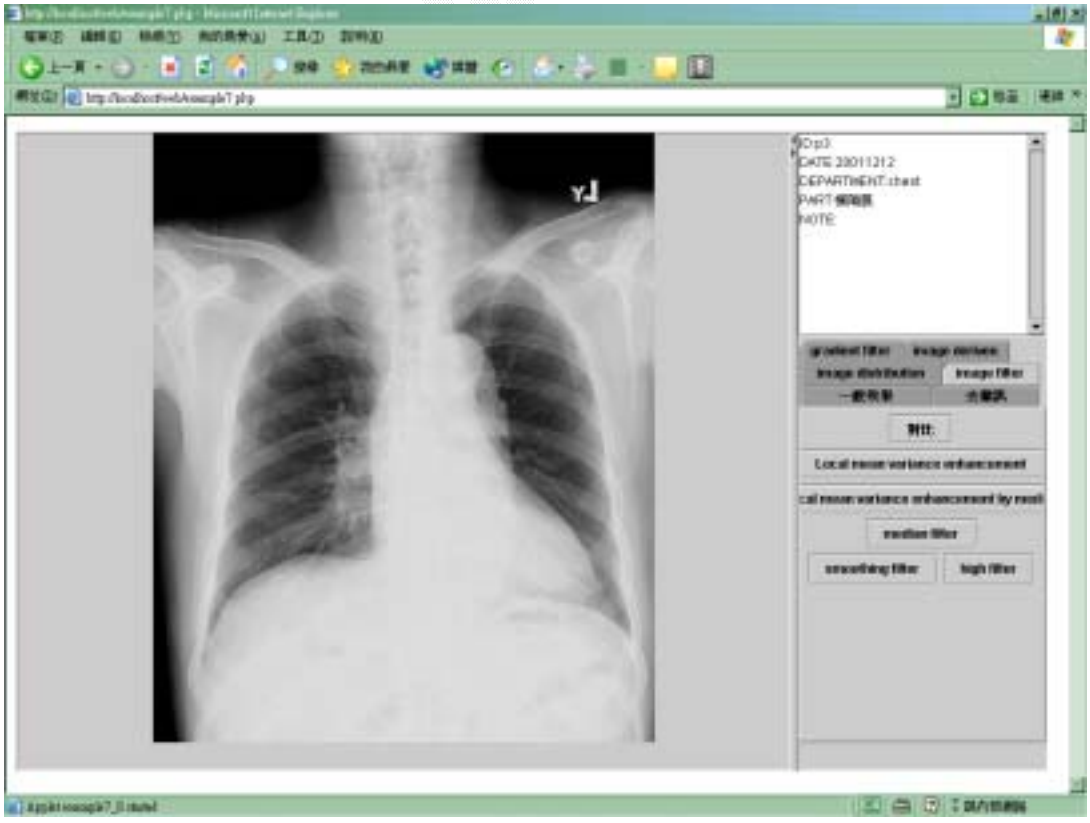
圖十七



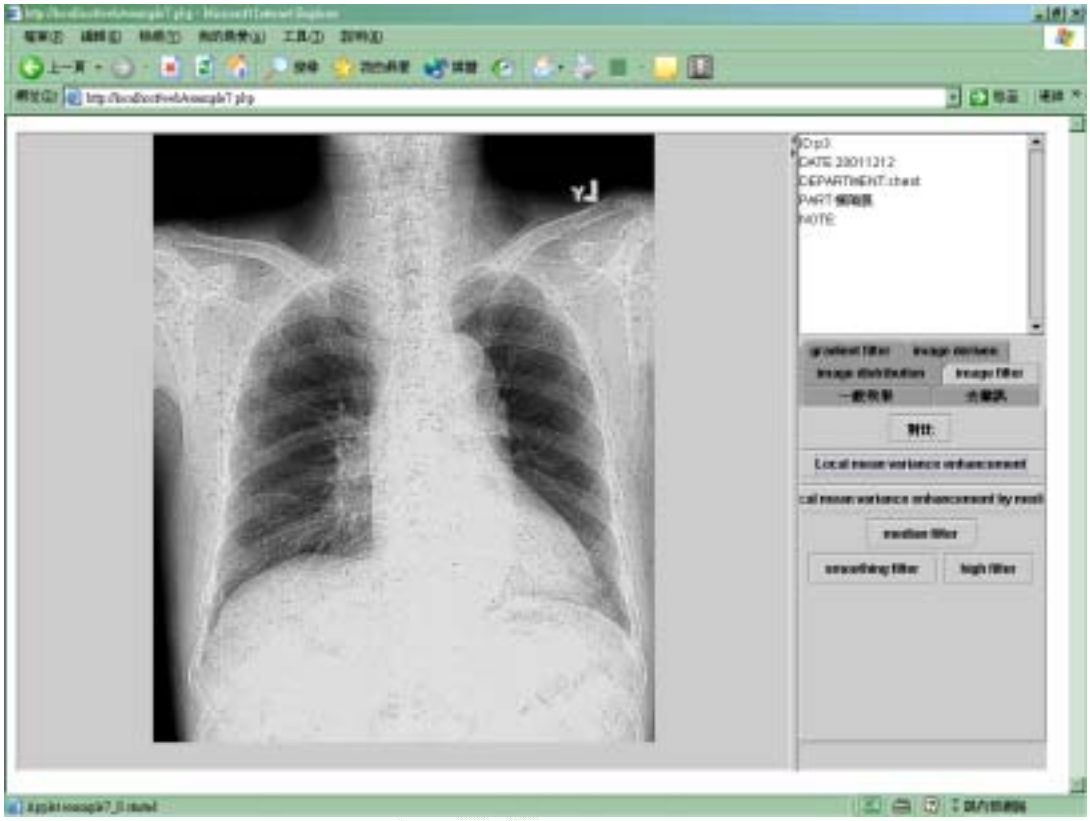
圖十八



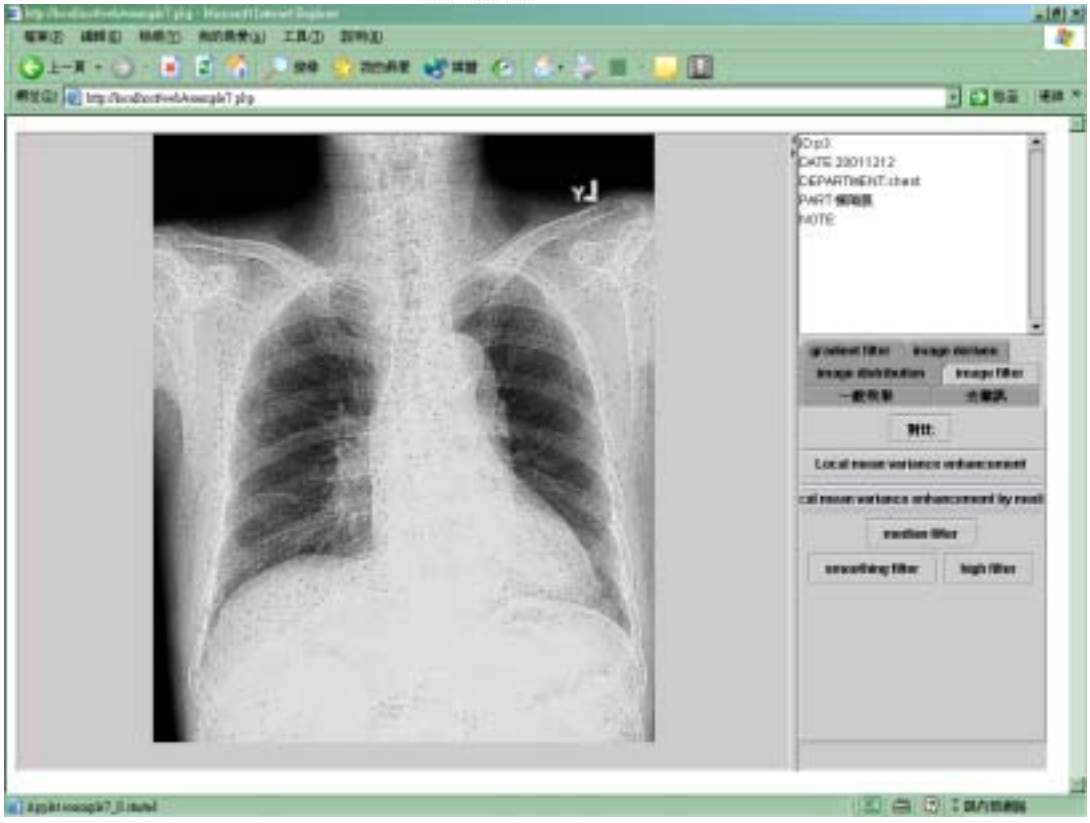
圖十九



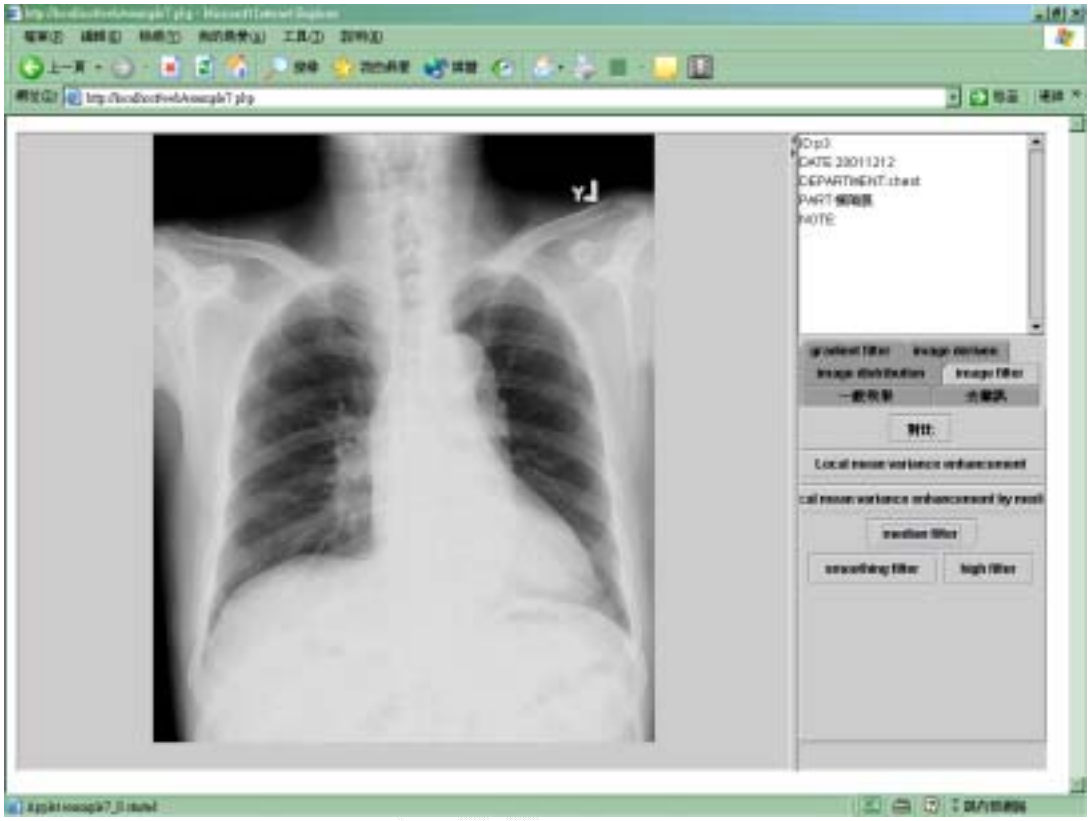
圖二十



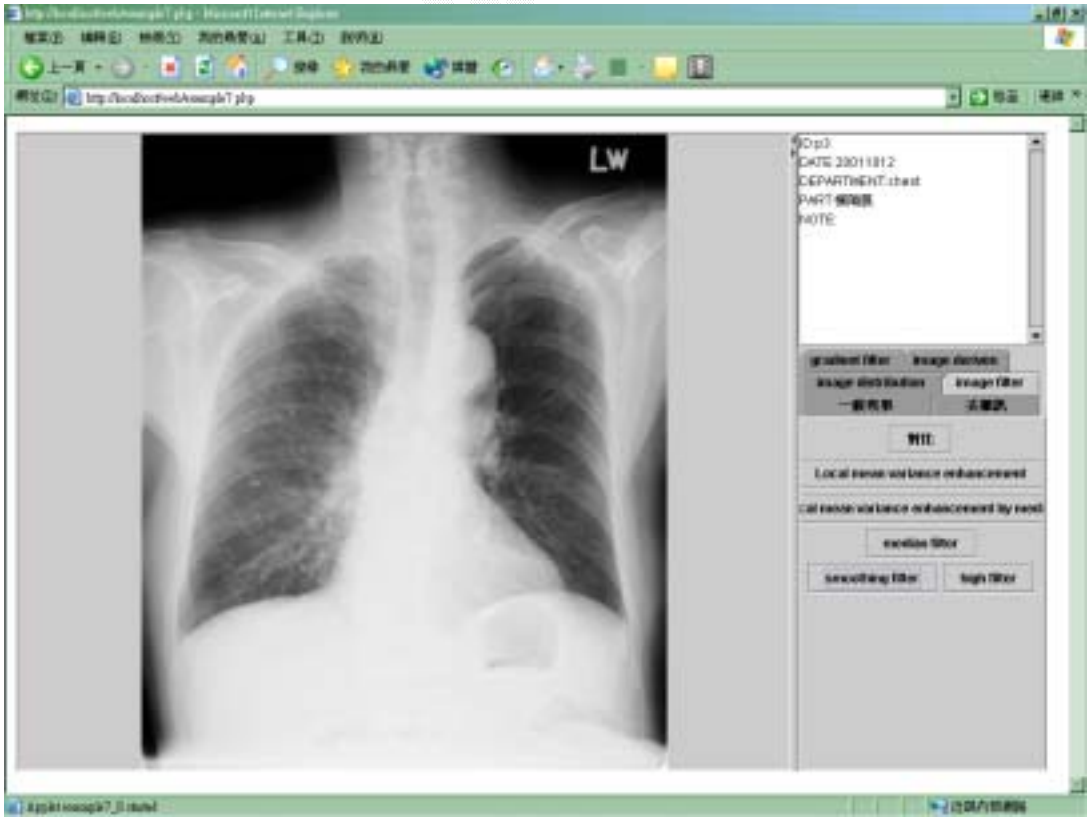
圖二十一



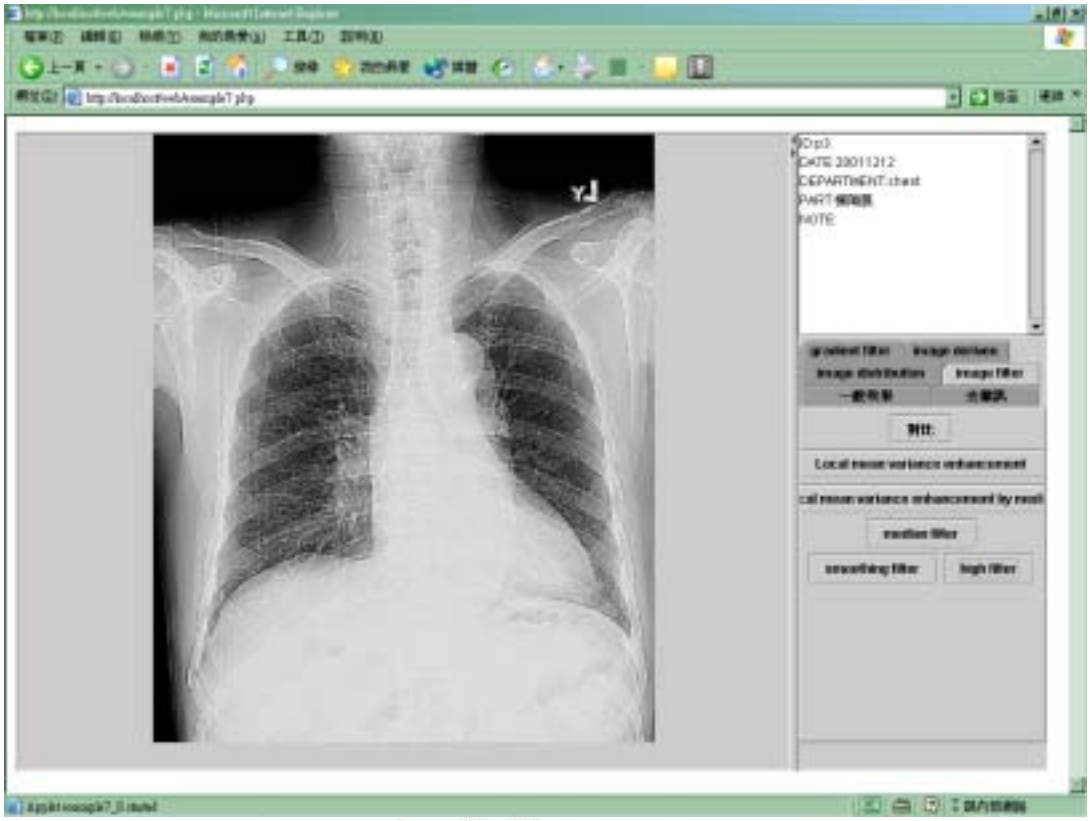
圖二十二



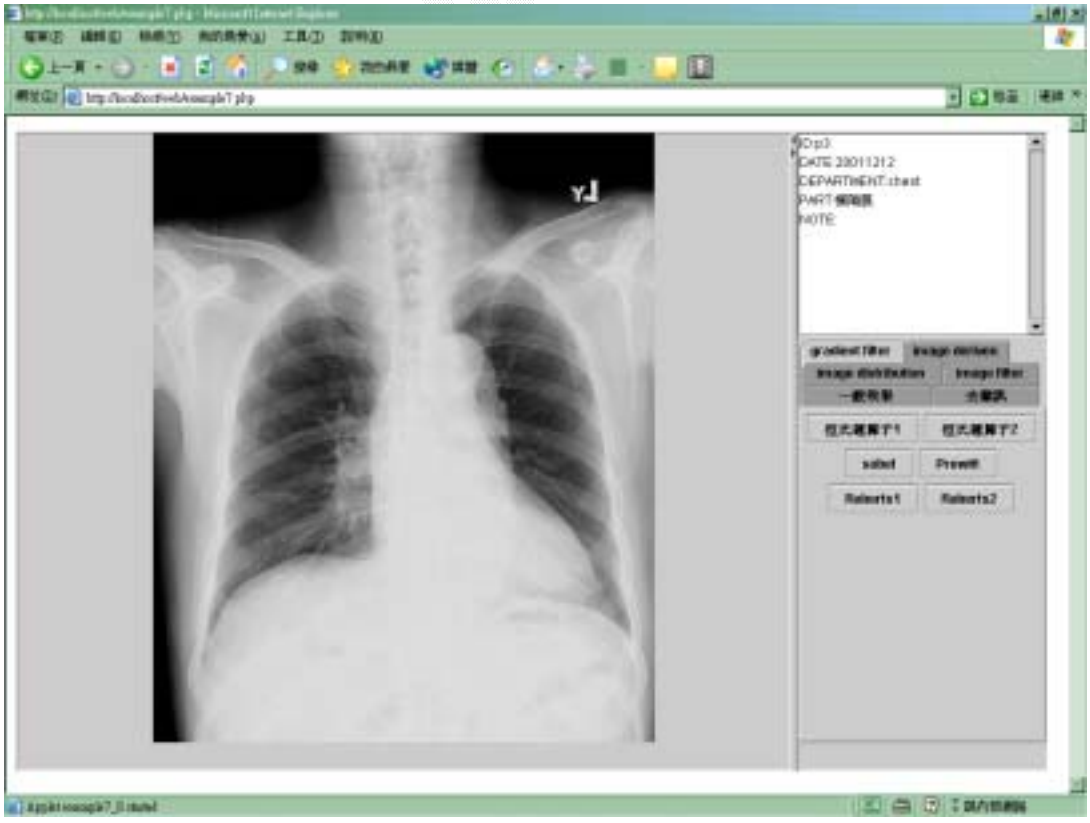
圖二十三



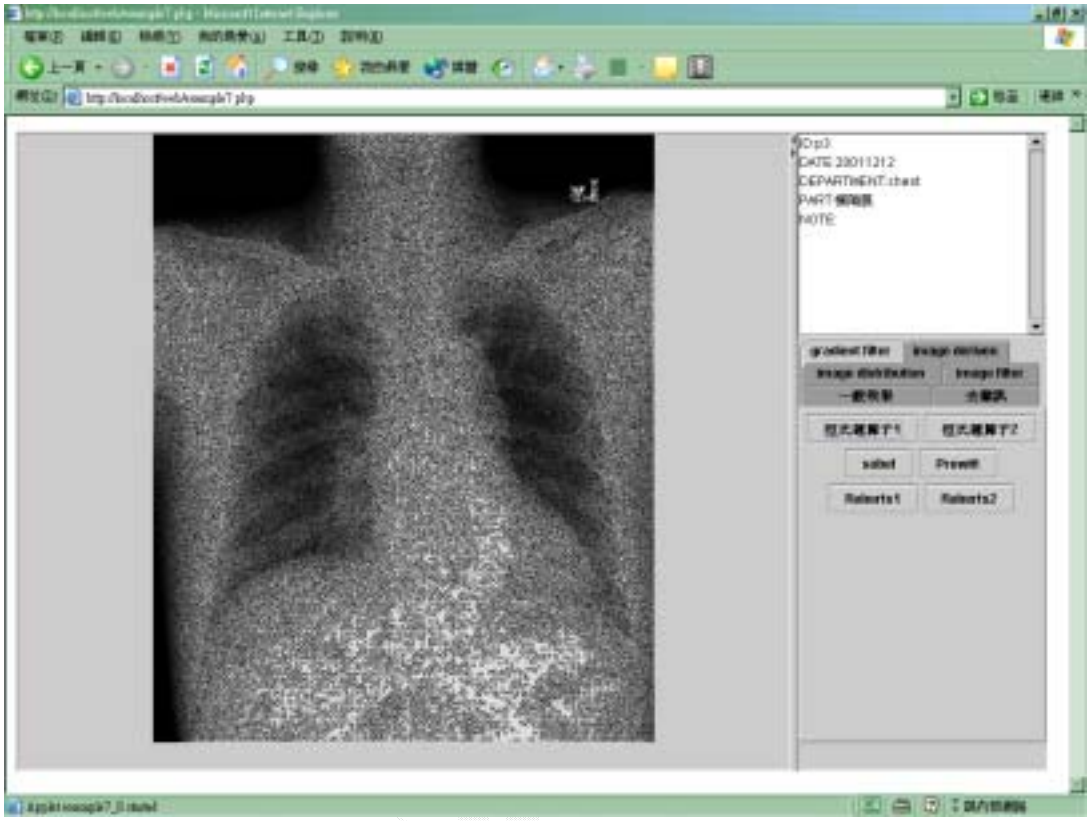
圖二十四



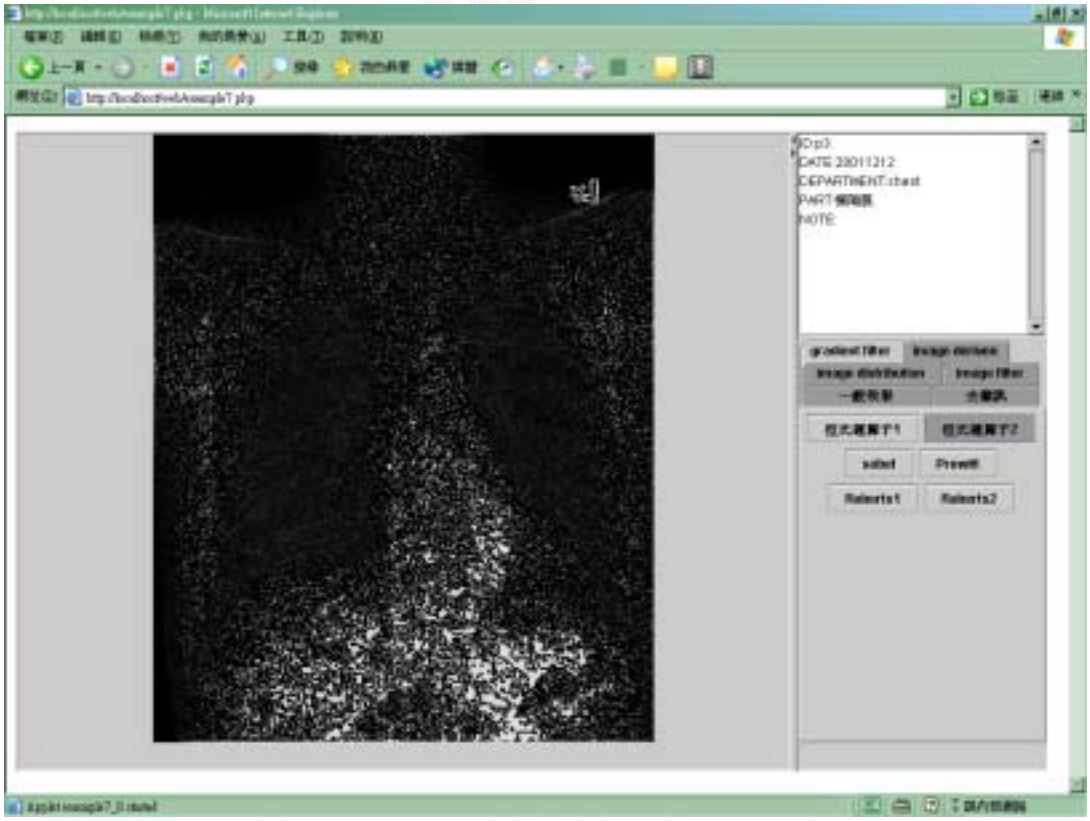
圖二十五



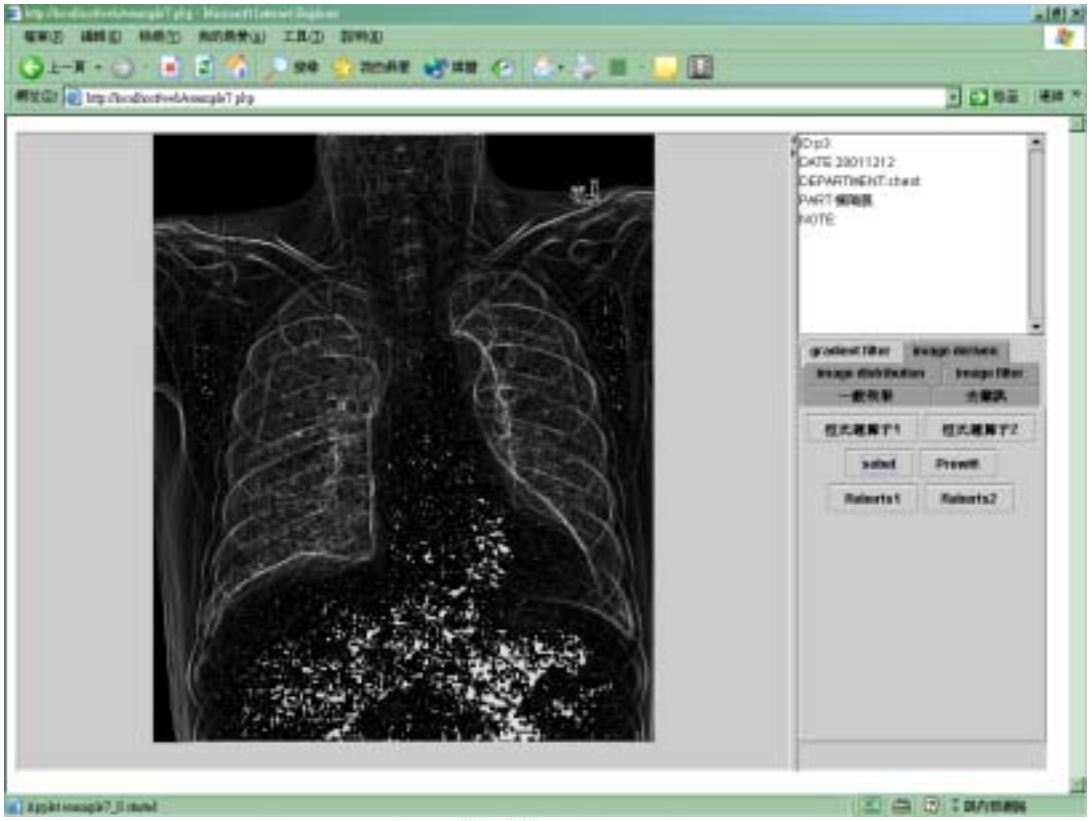
圖二十六



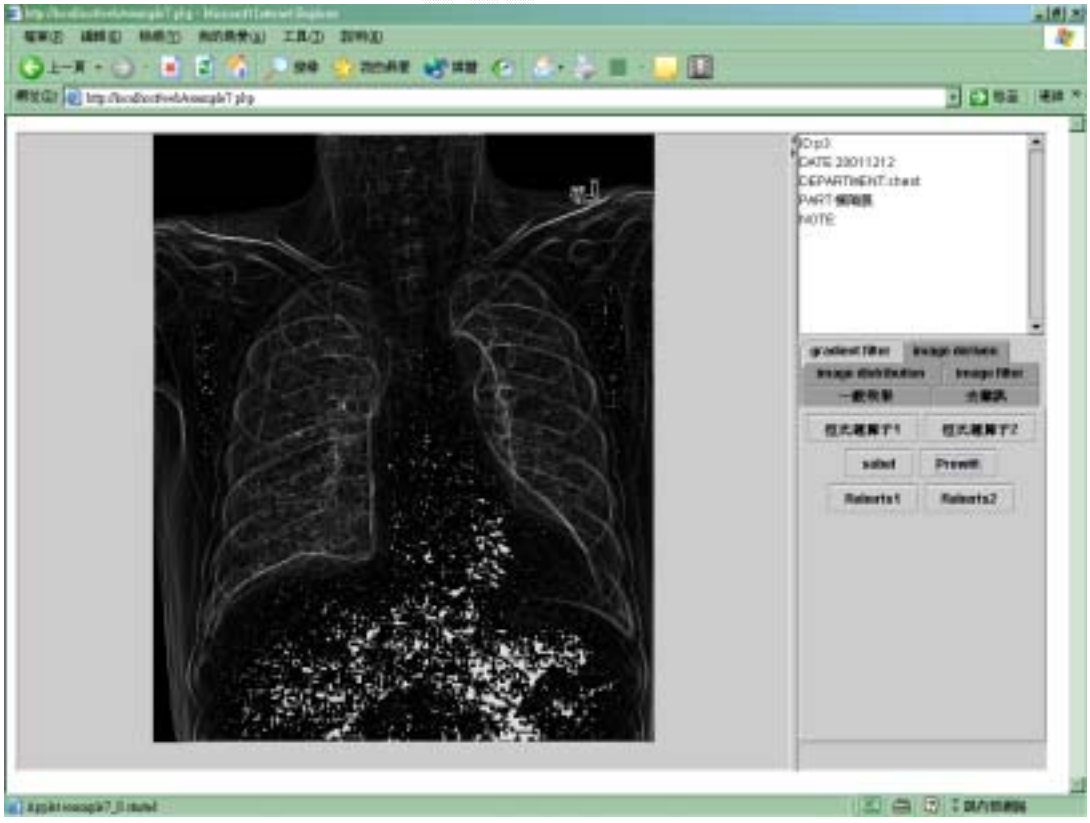
圖二十七



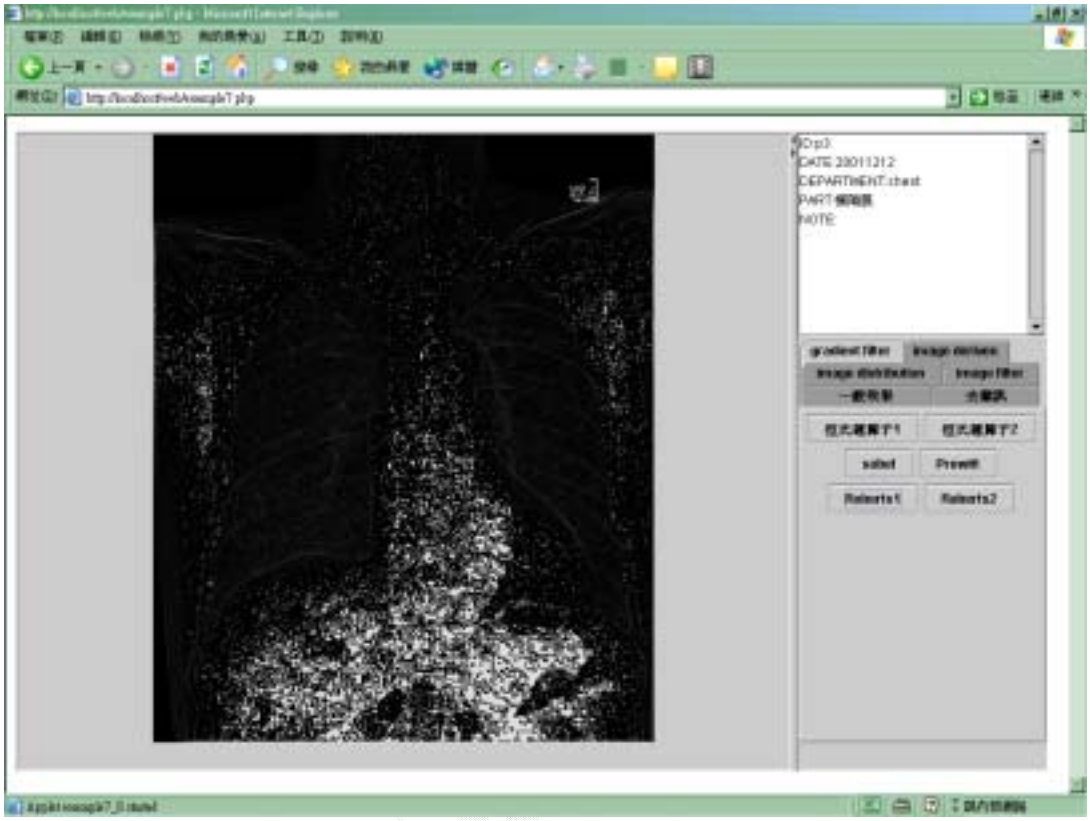
圖二十八



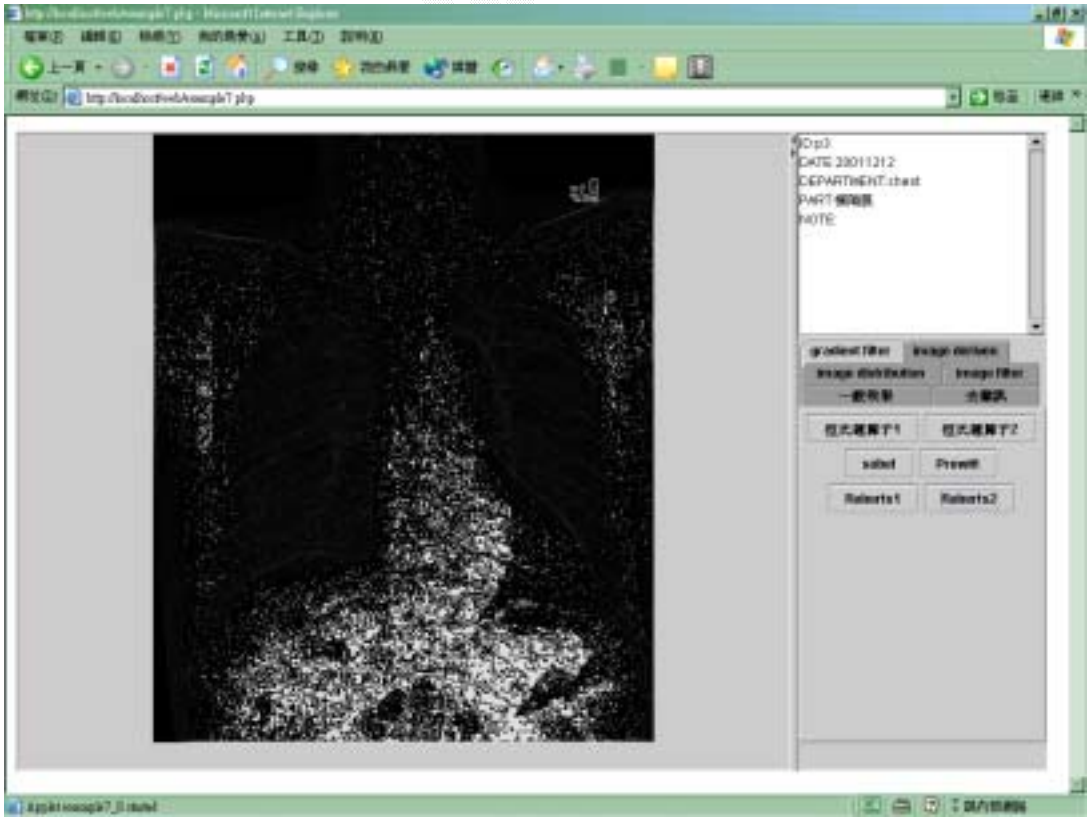
圖二十九



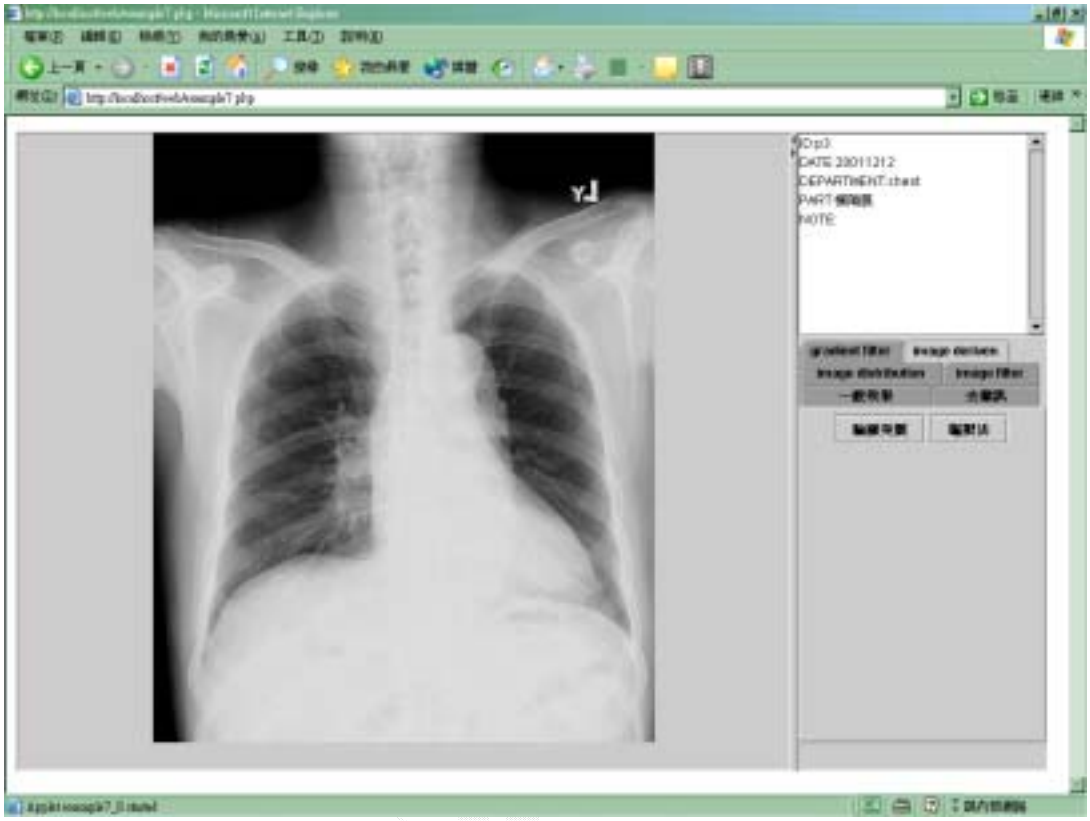
圖三十



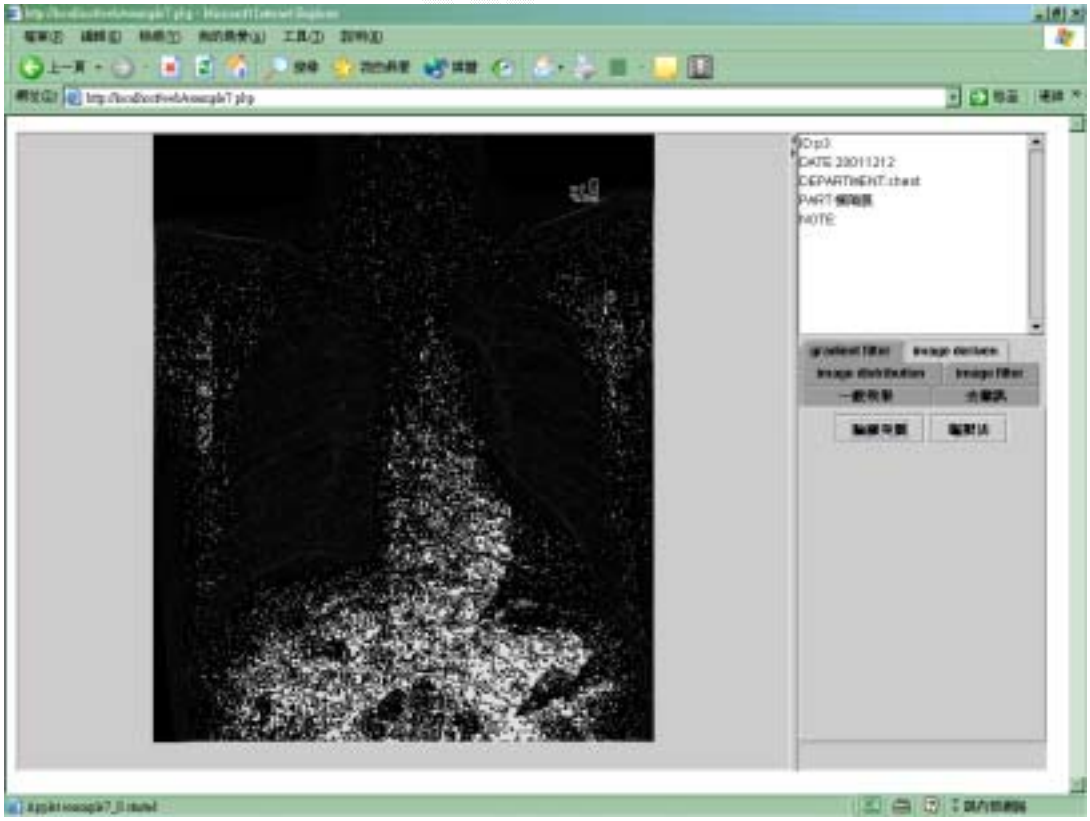
圖三十一



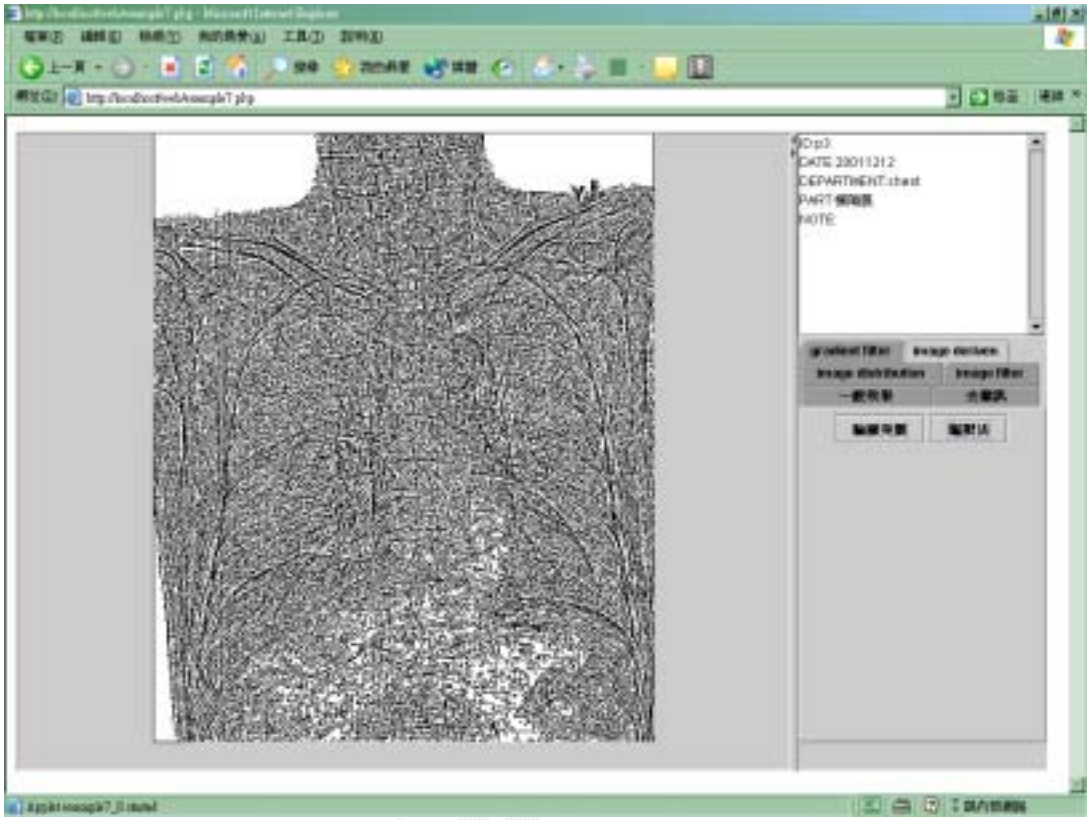
圖三十二



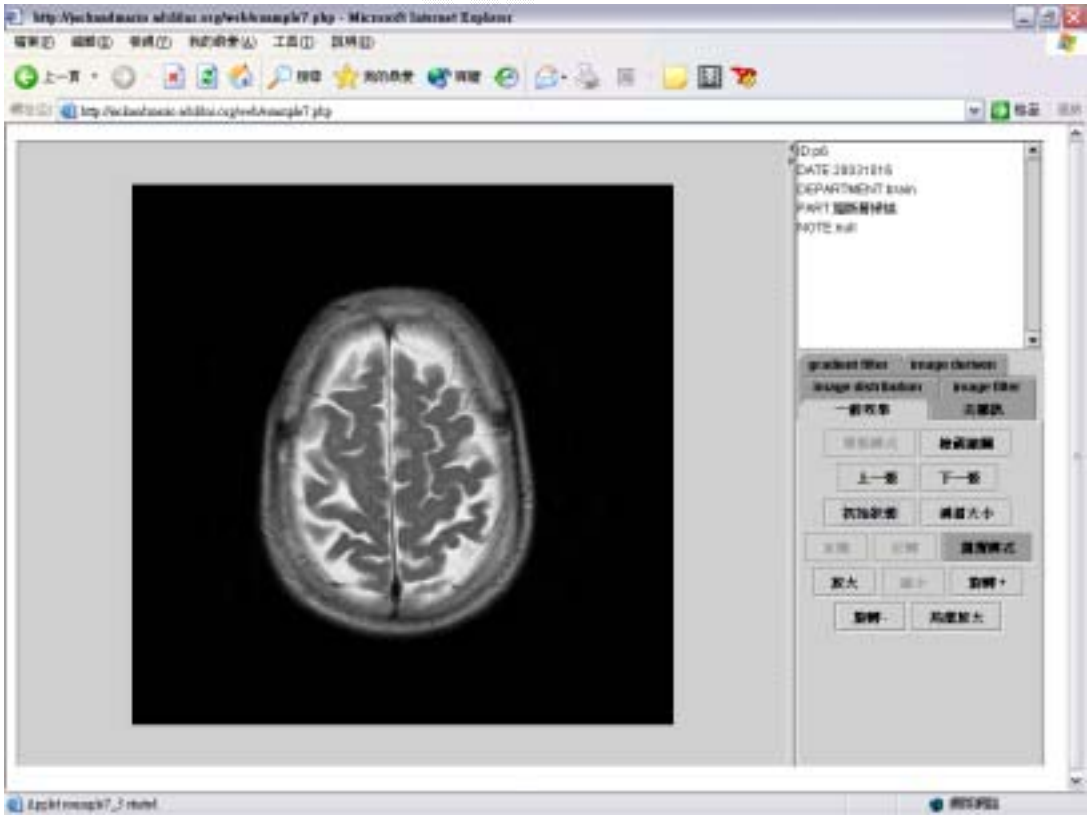
圖三十三



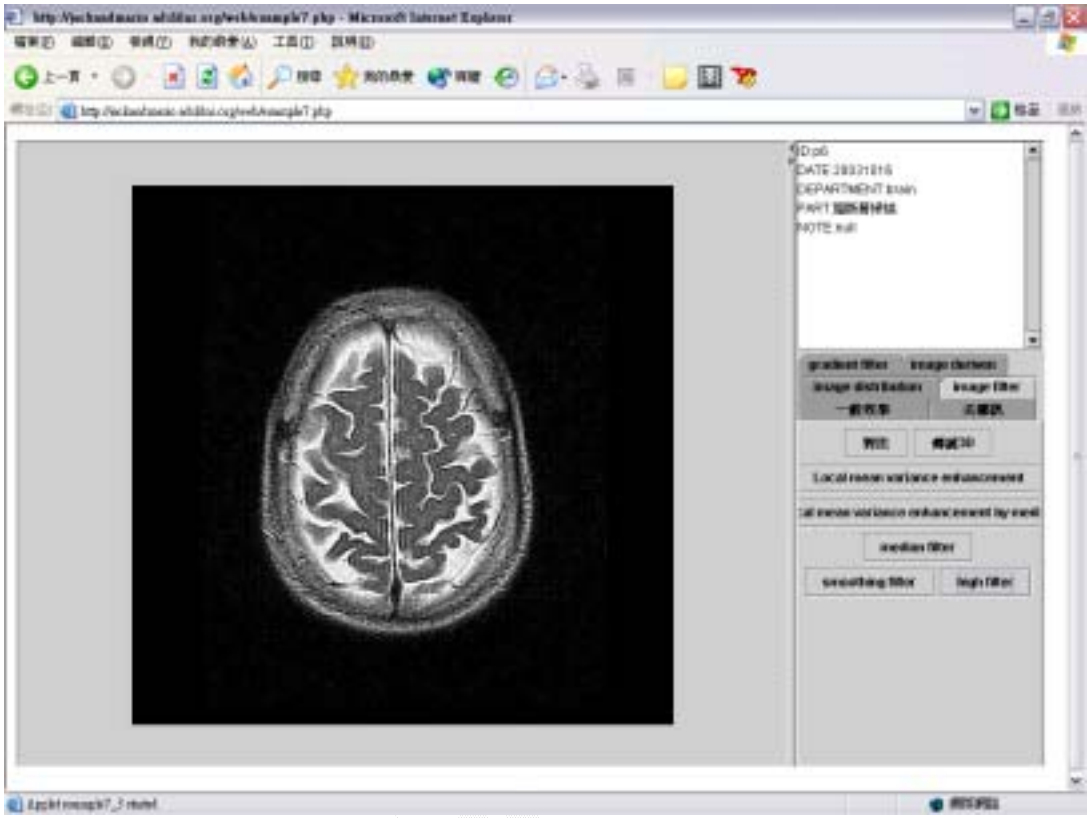
圖三十四



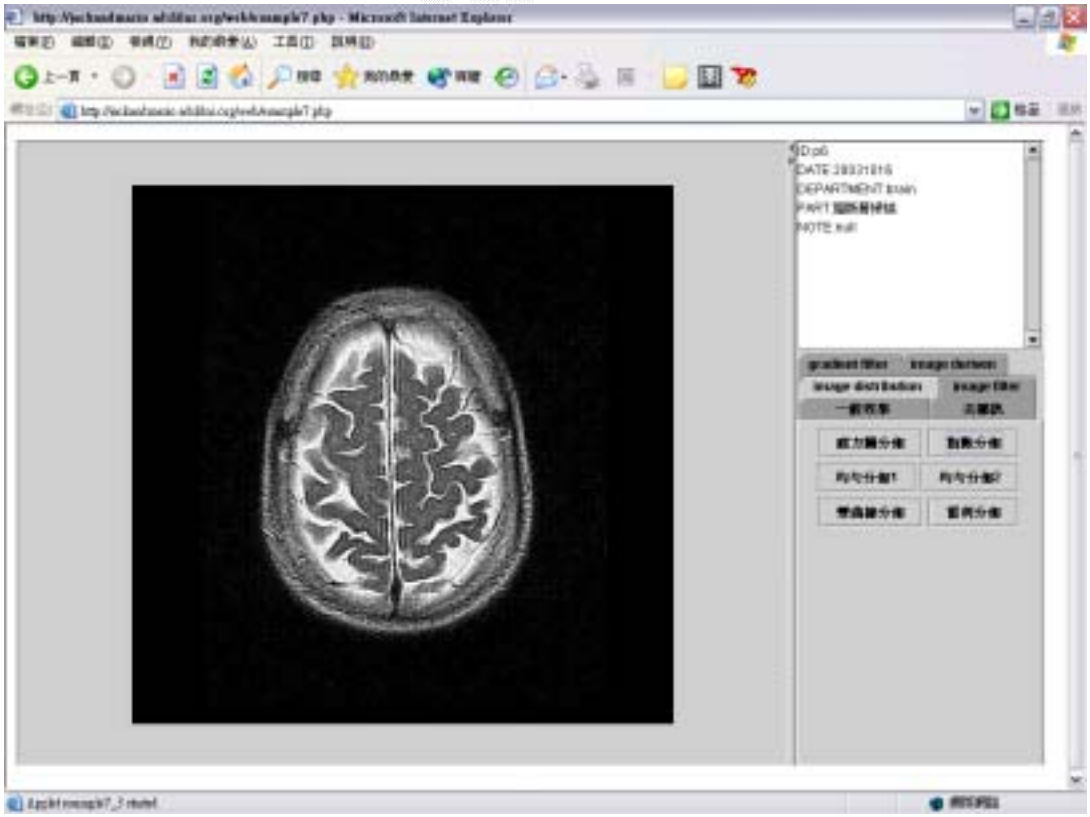
圖三十五



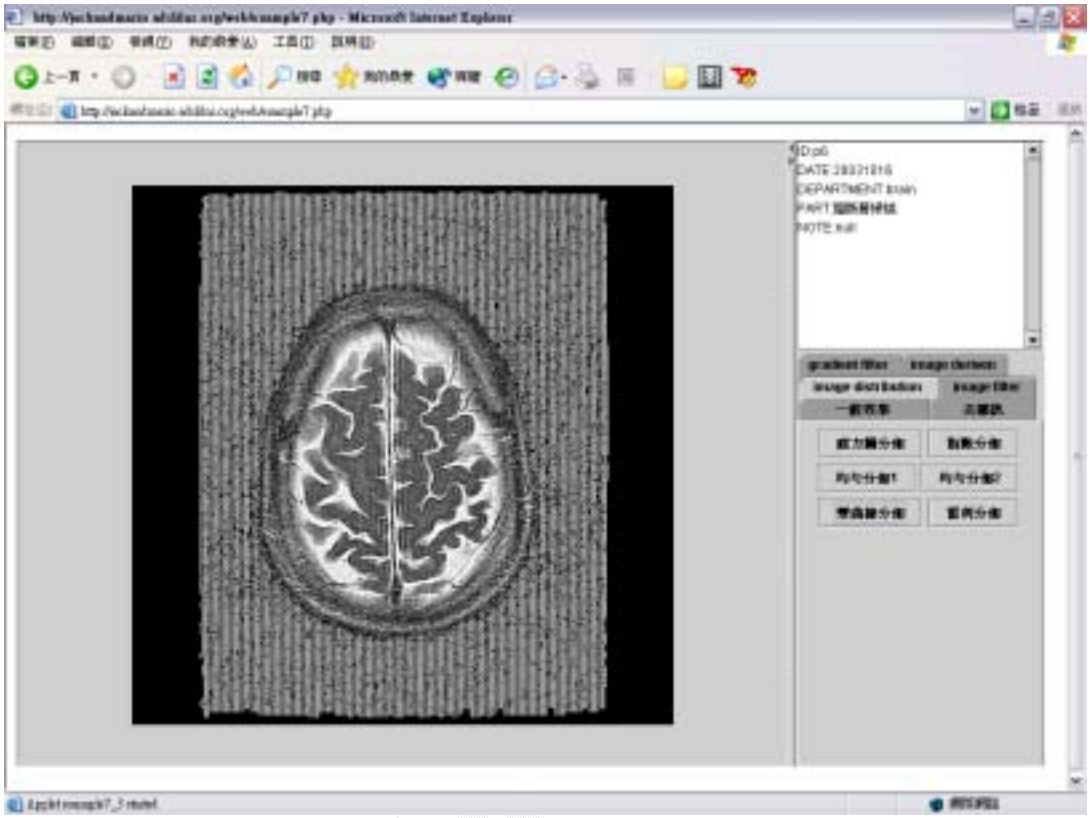
圖三十六



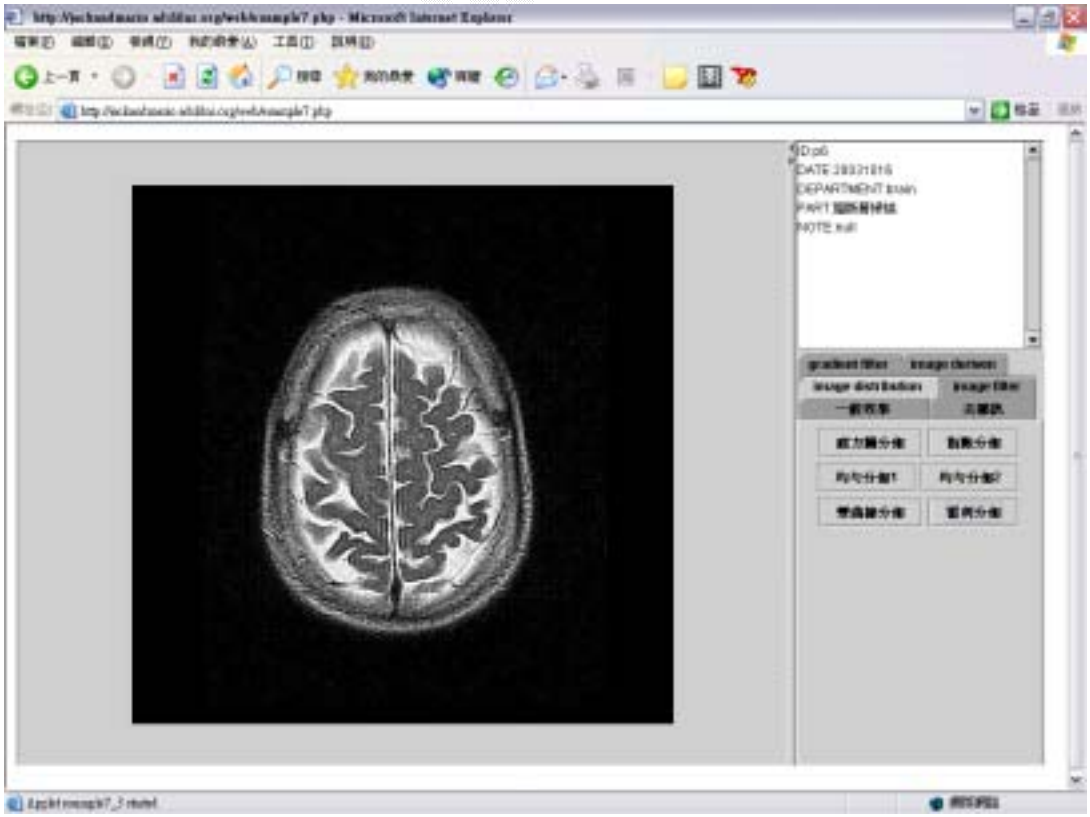
圖三十七



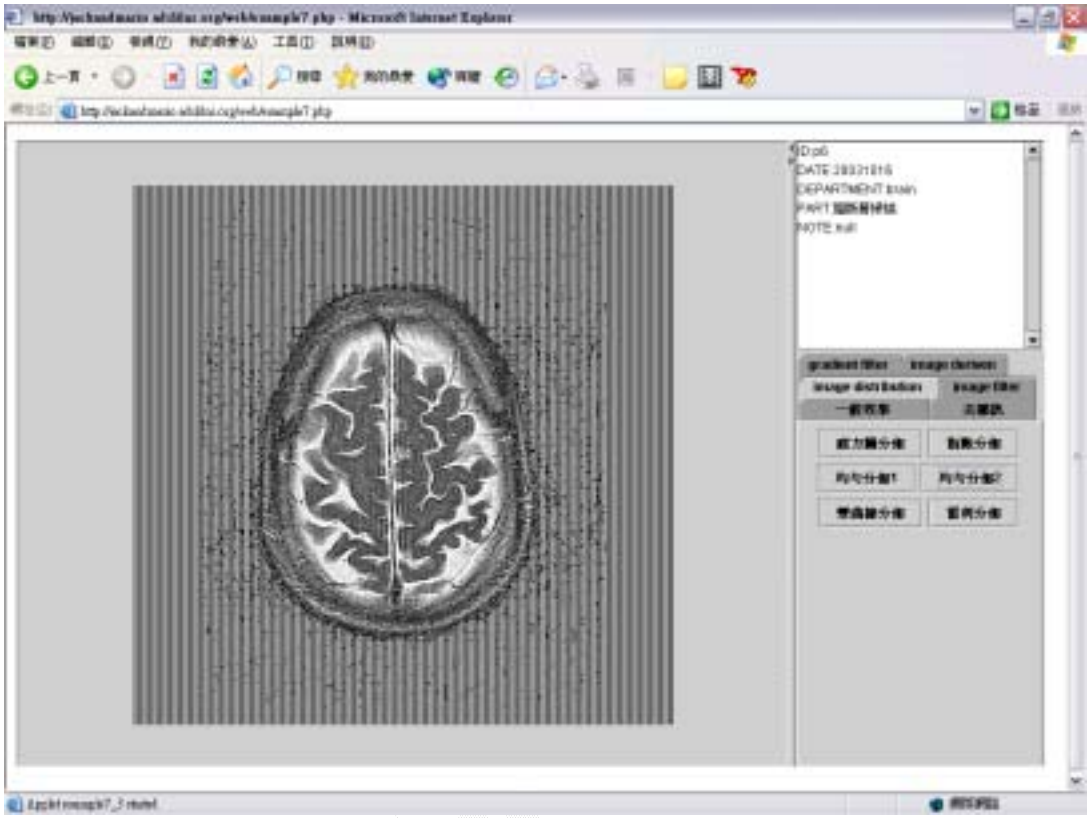
圖三十八



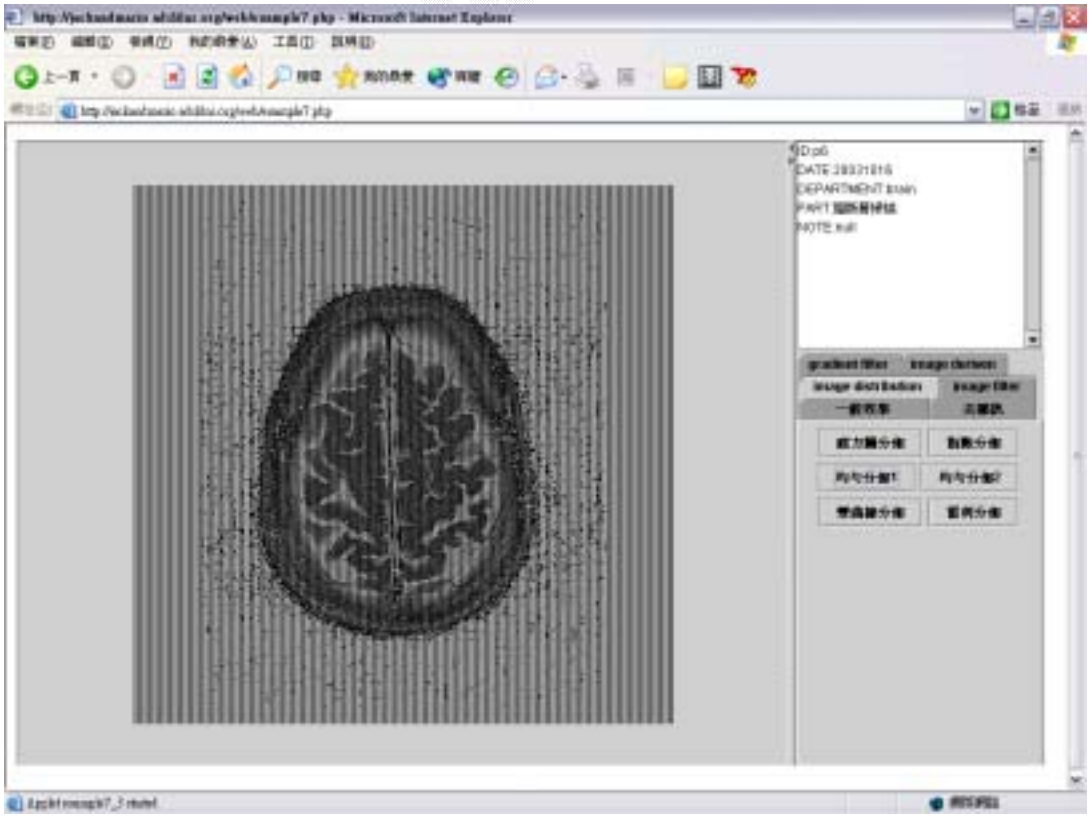
圖三十九



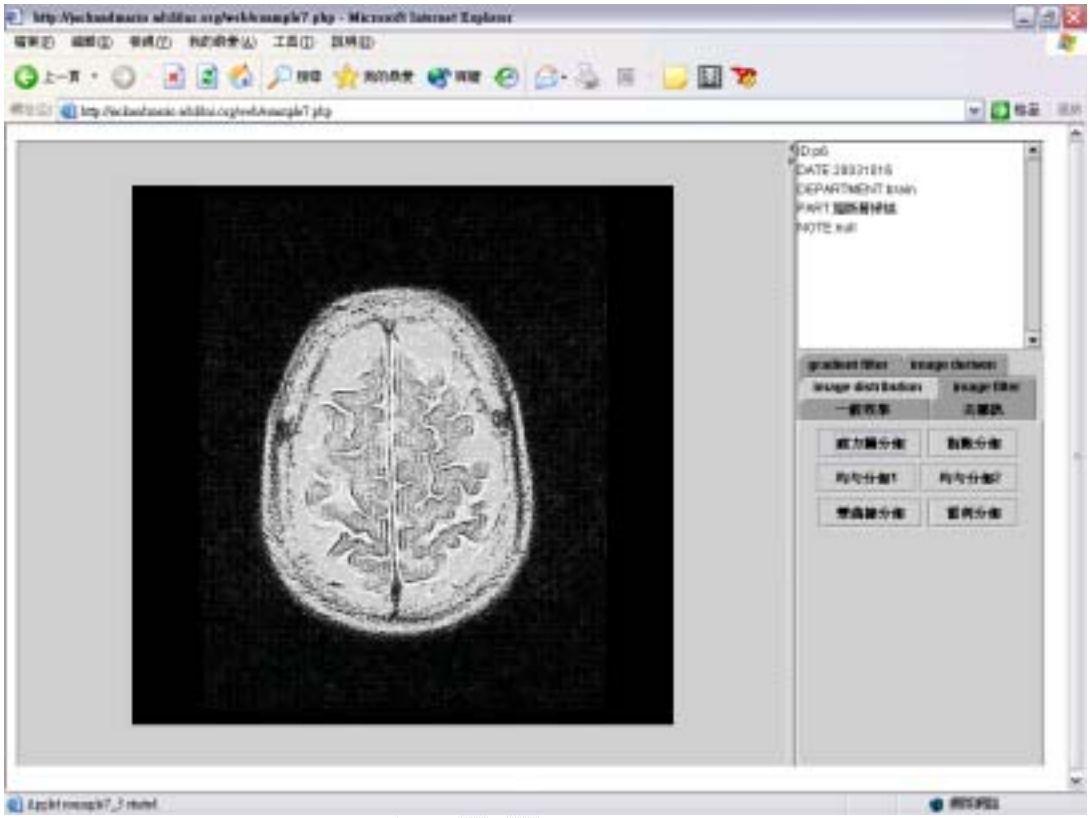
圖四十



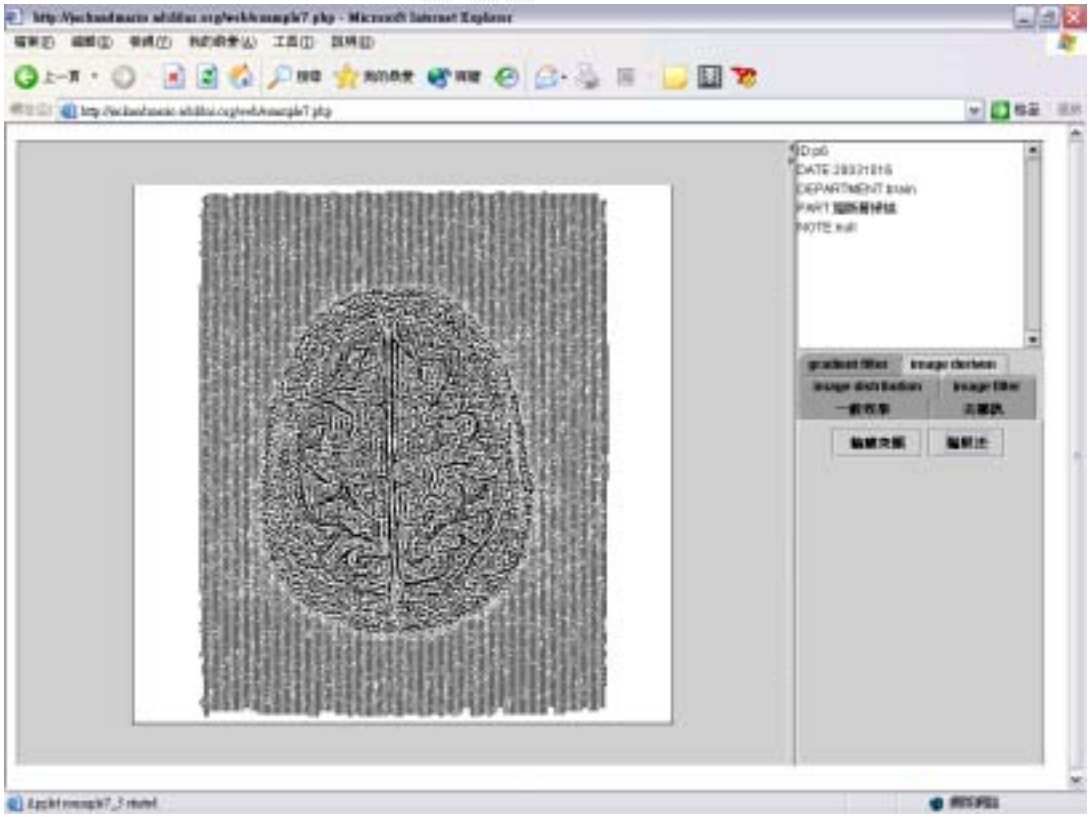
圖四十一



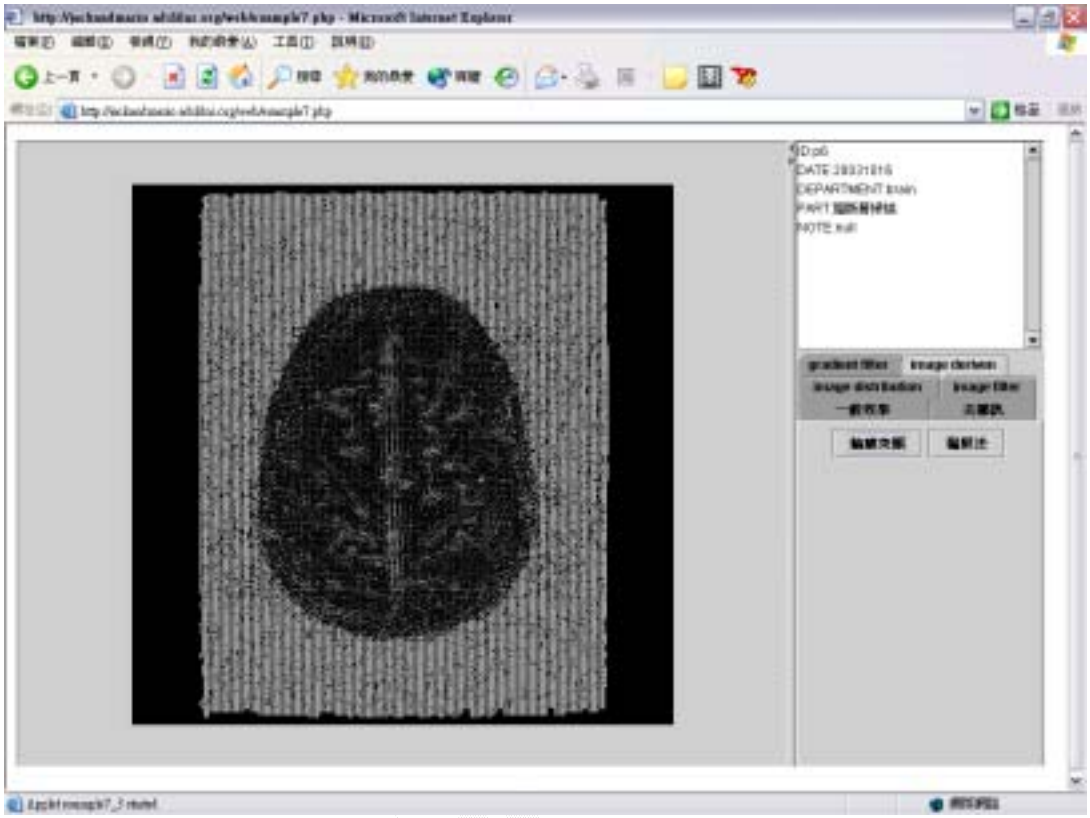
圖四十二



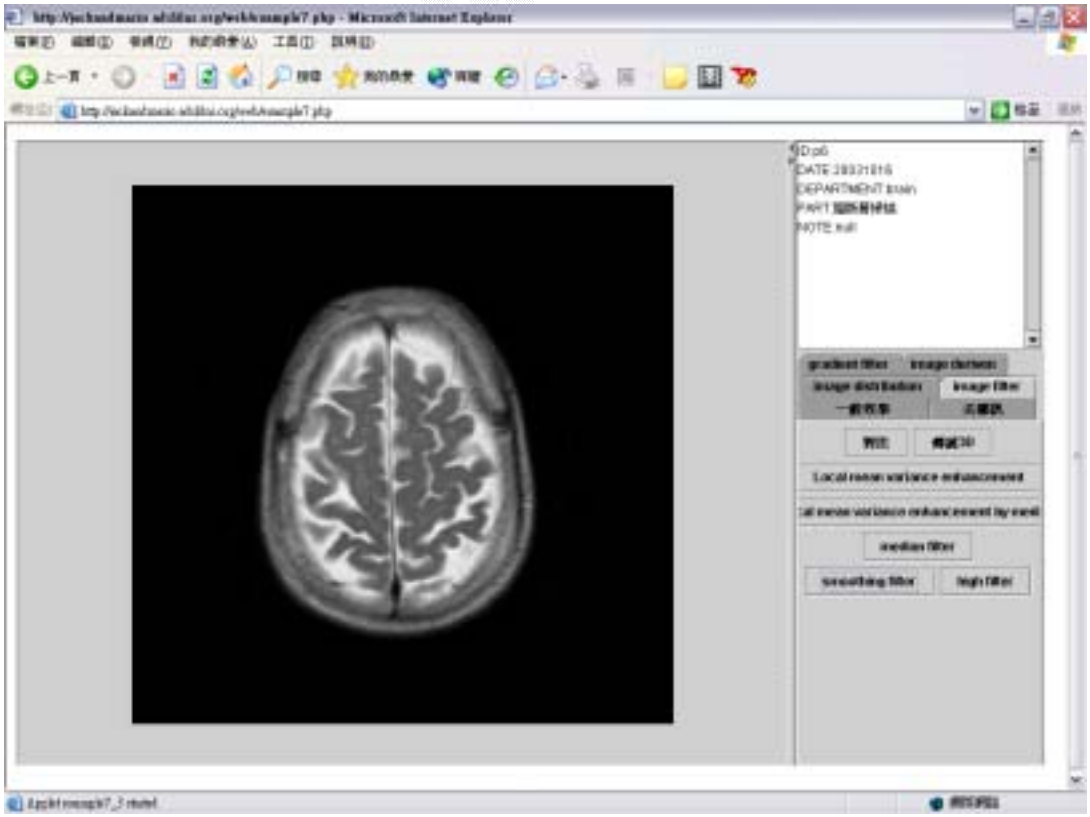
圖四十三



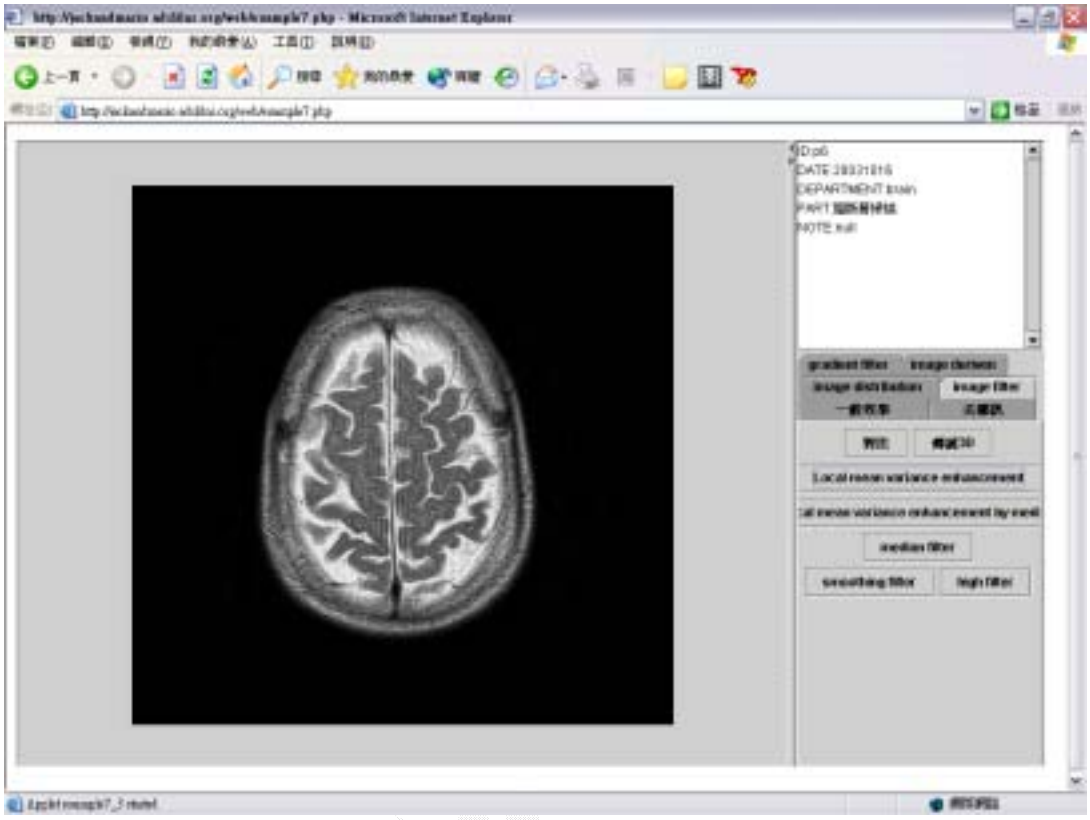
圖四十四



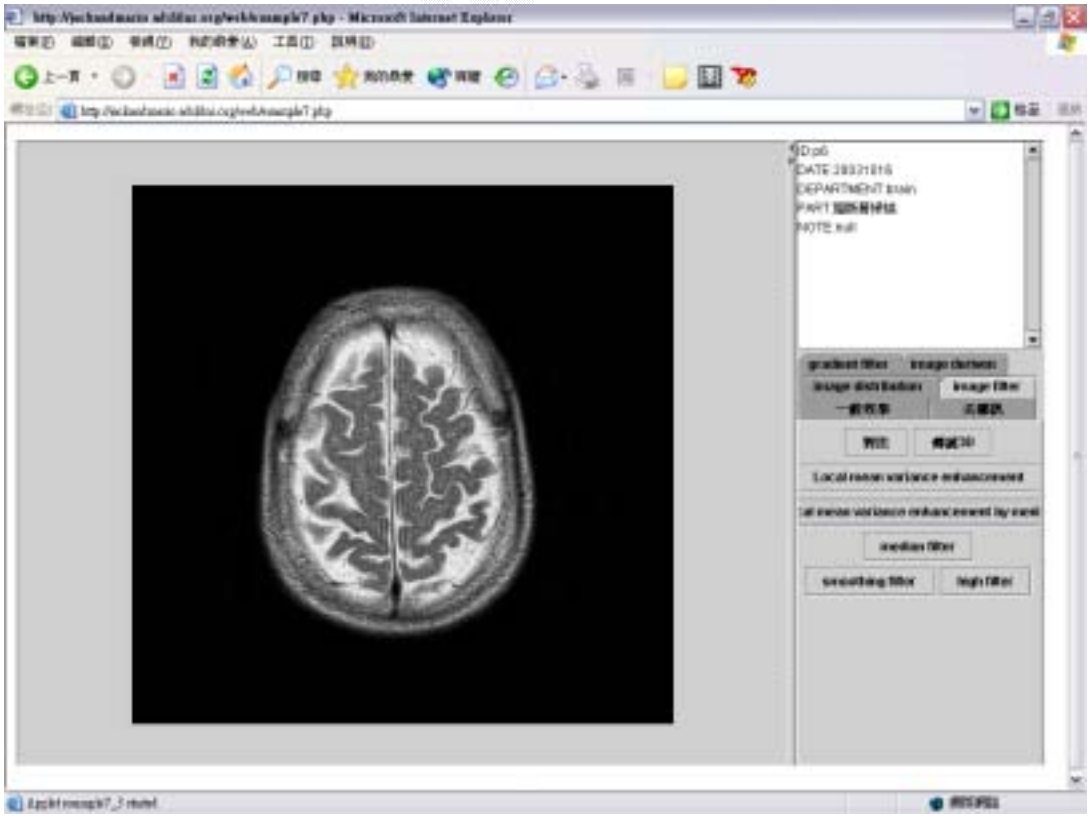
圖四十五



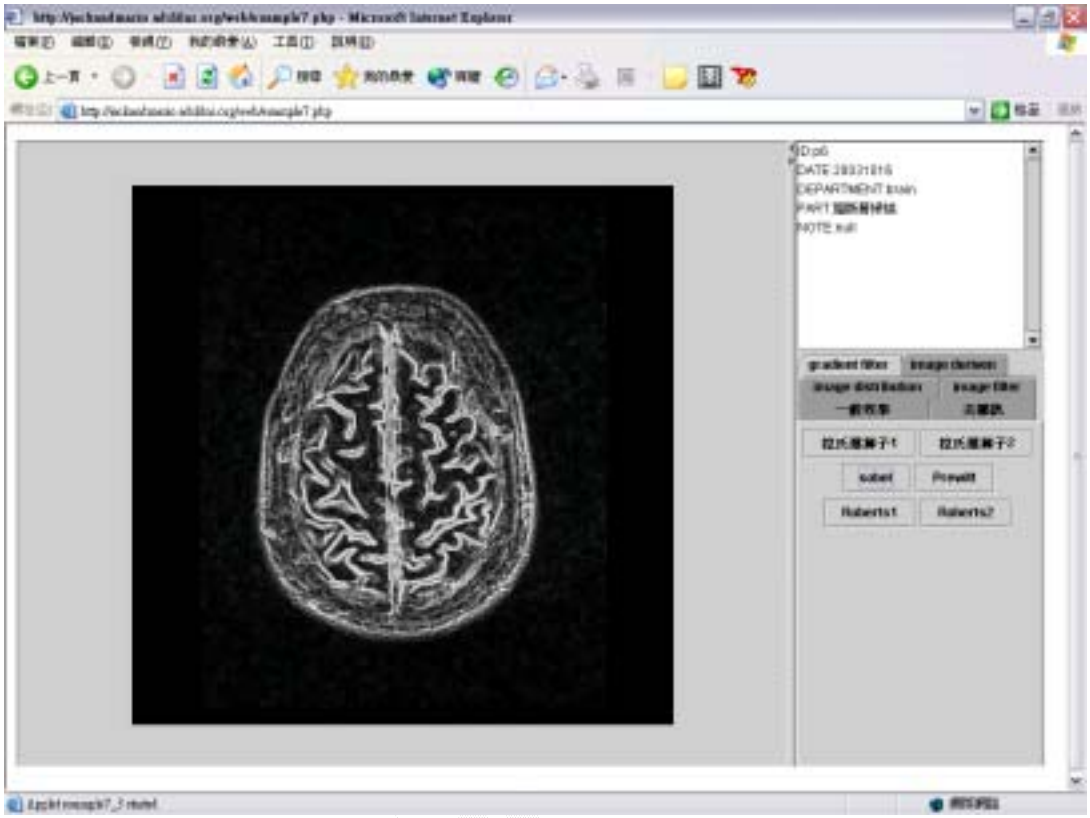
圖四十六



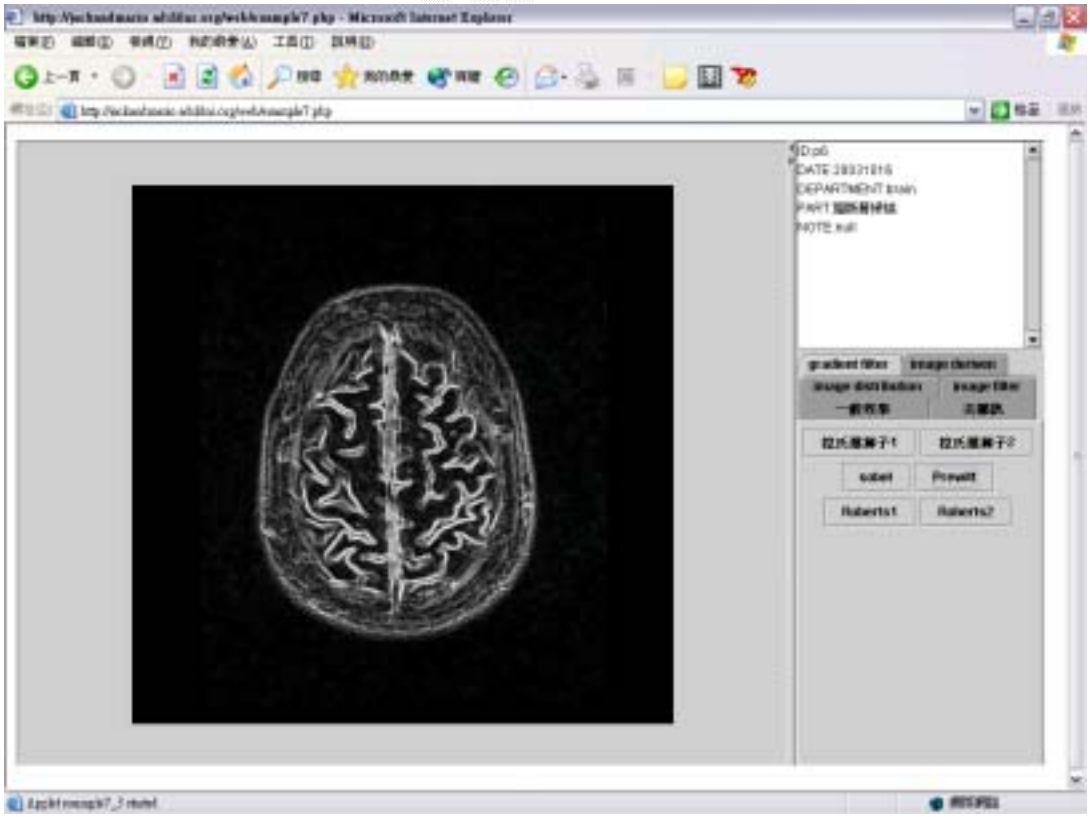
圖四十七



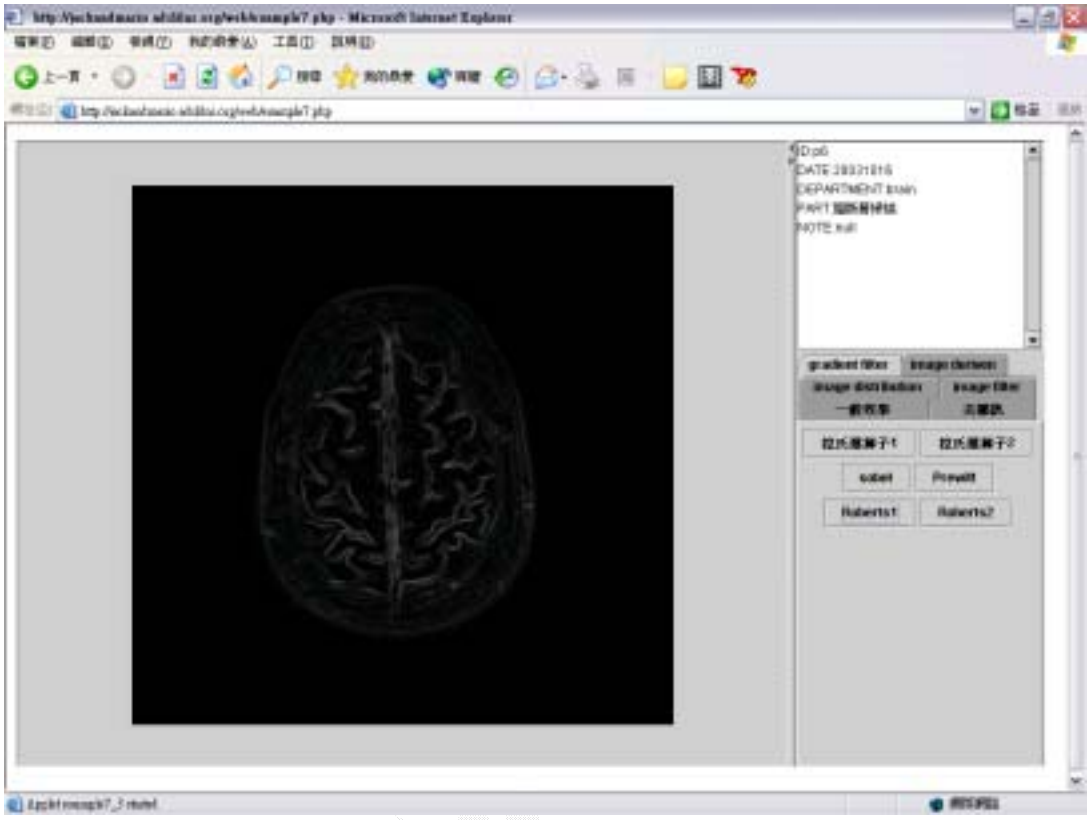
圖四十八



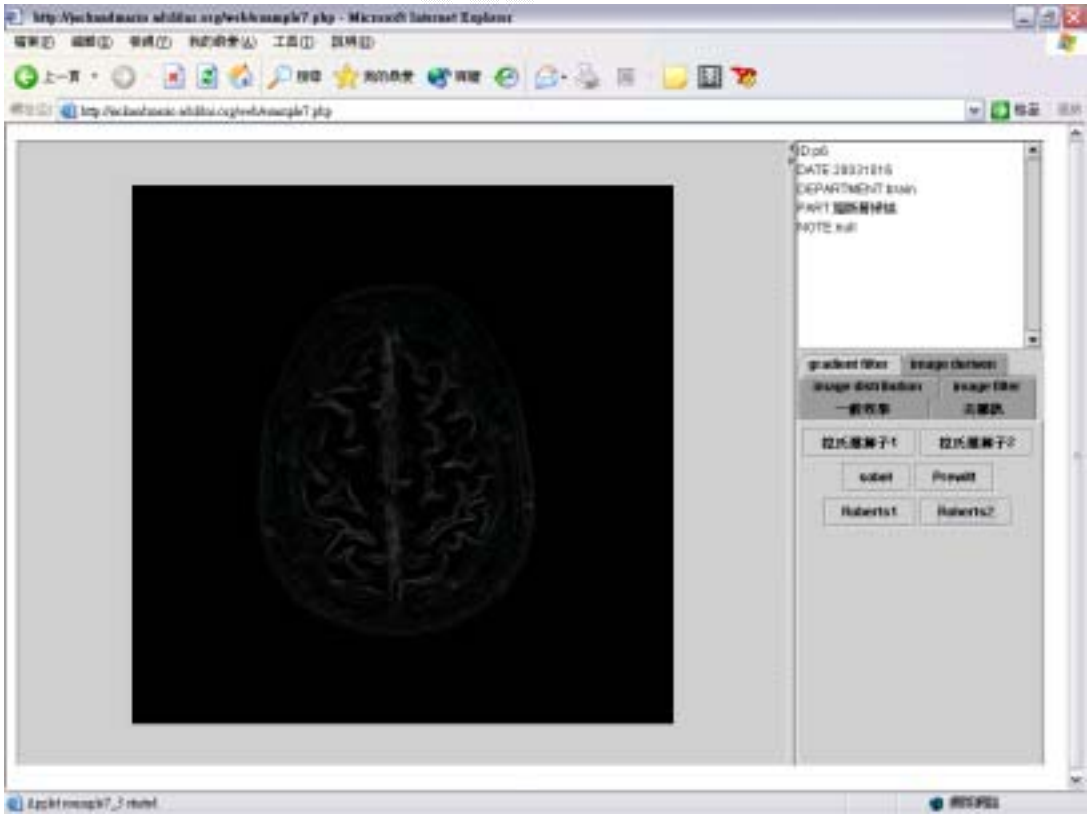
圖四十九



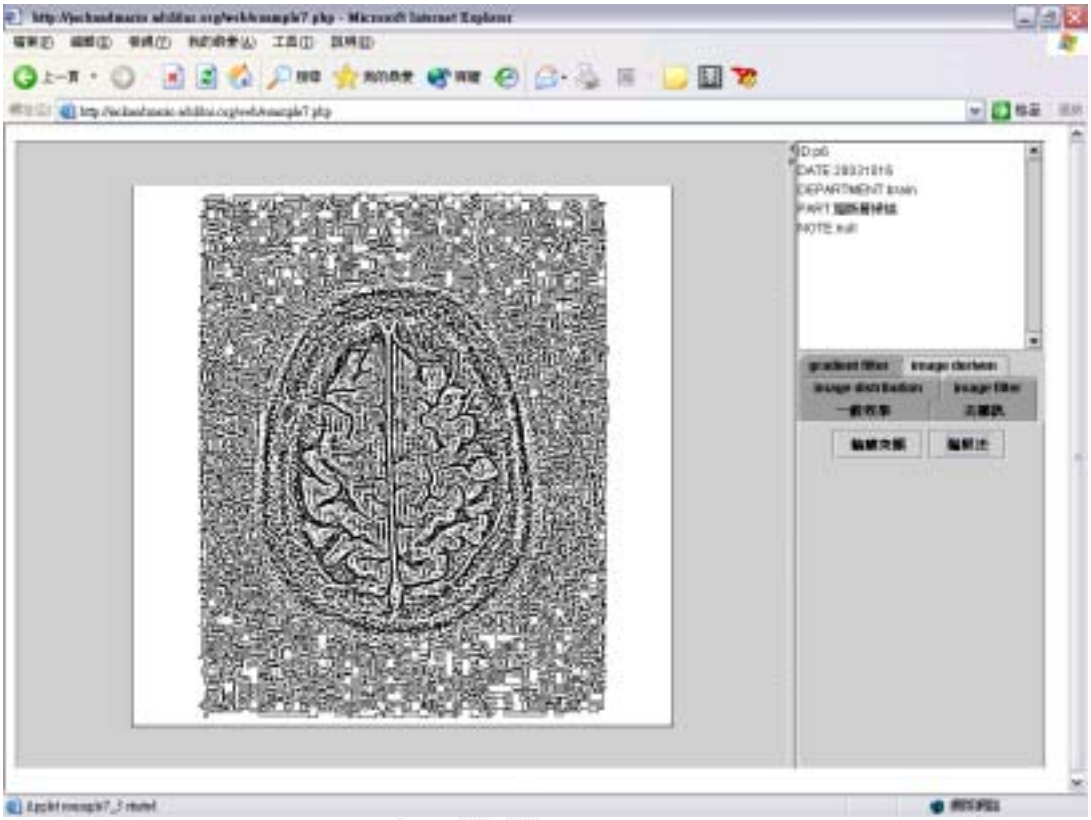
圖五十



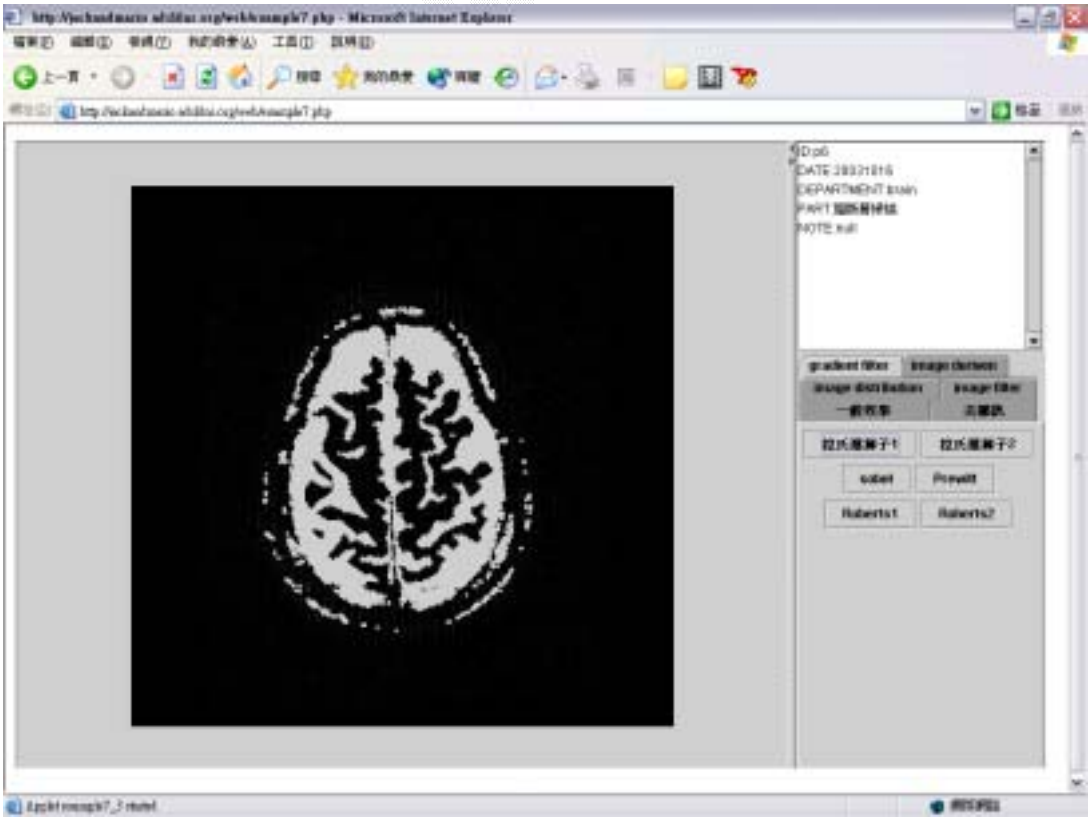
圖五十一



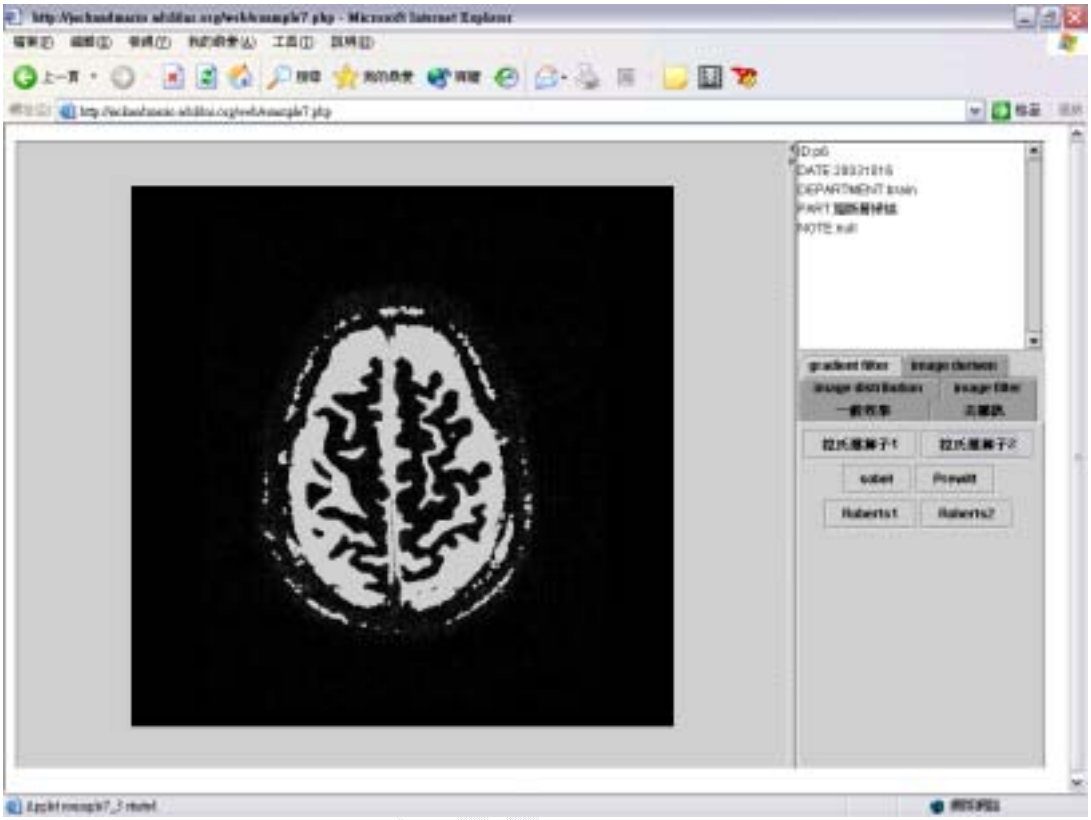
圖五十二



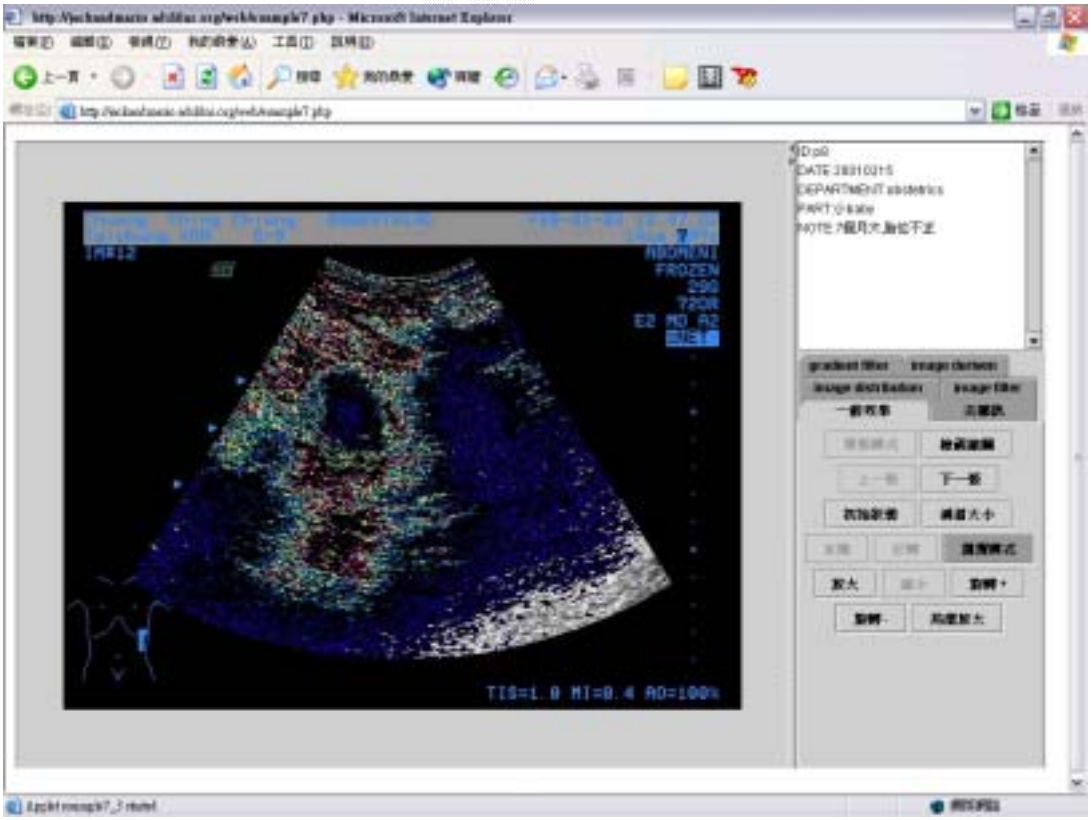
圖五十三



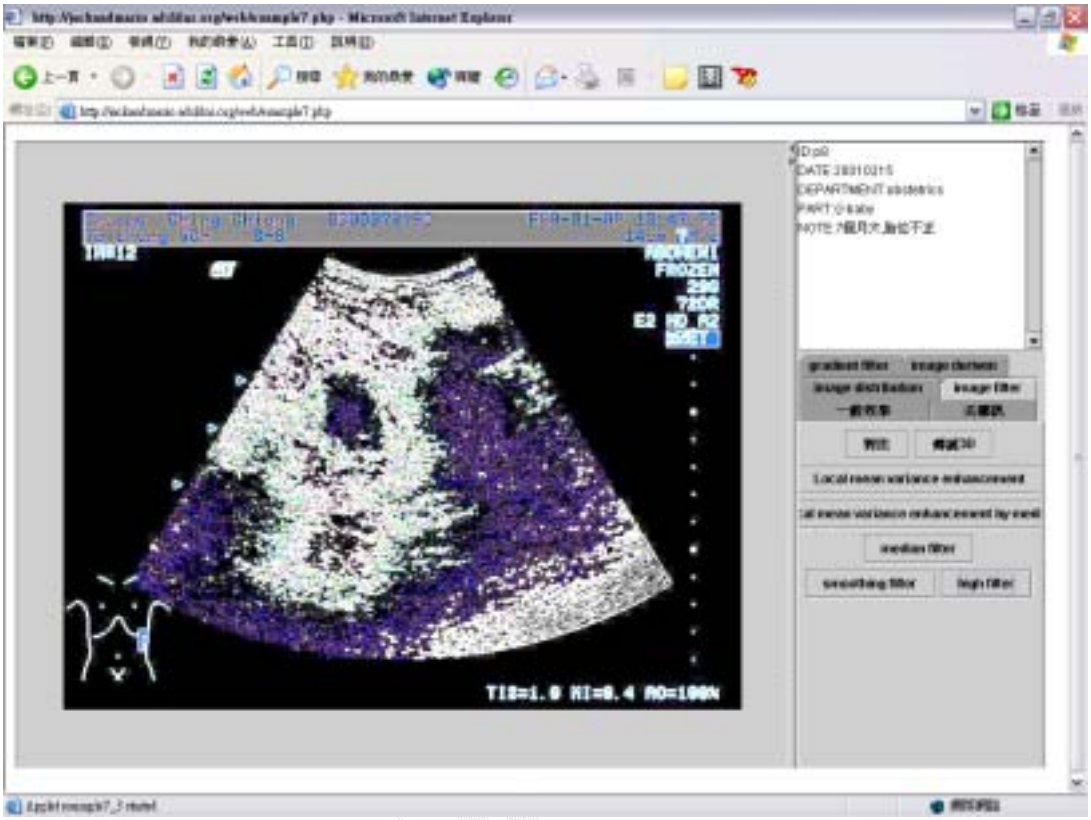
圖五十四



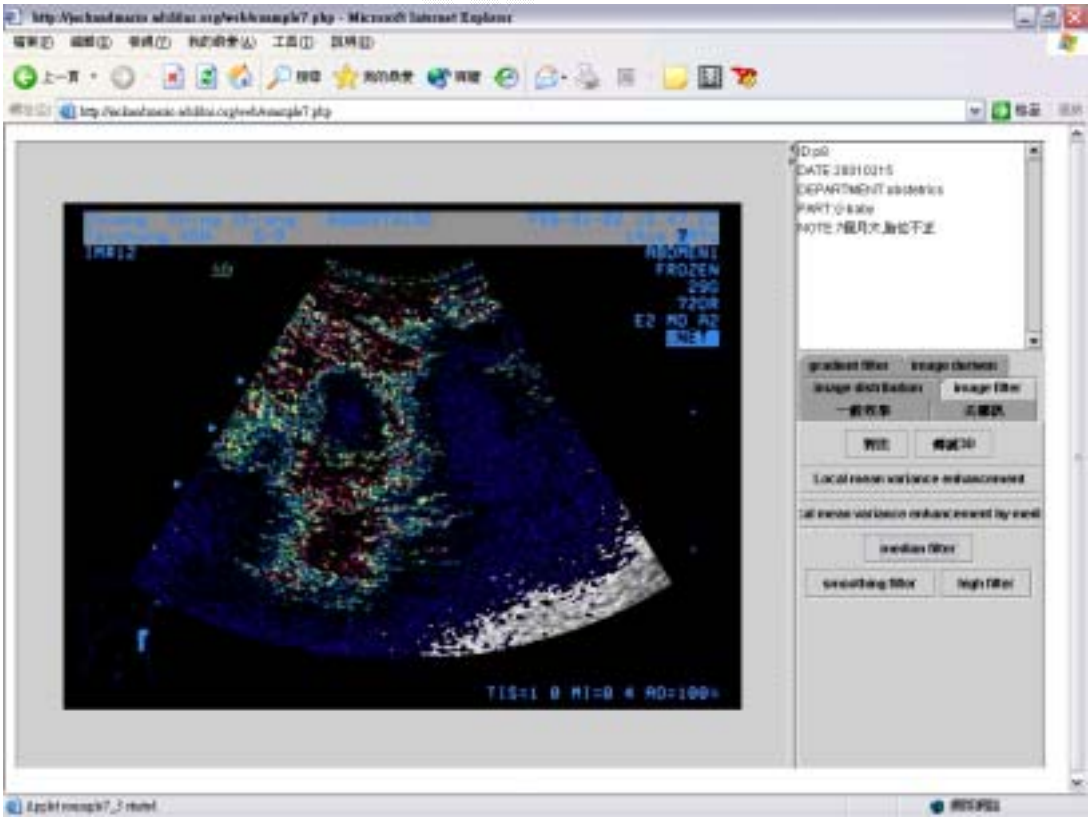
圖五十五



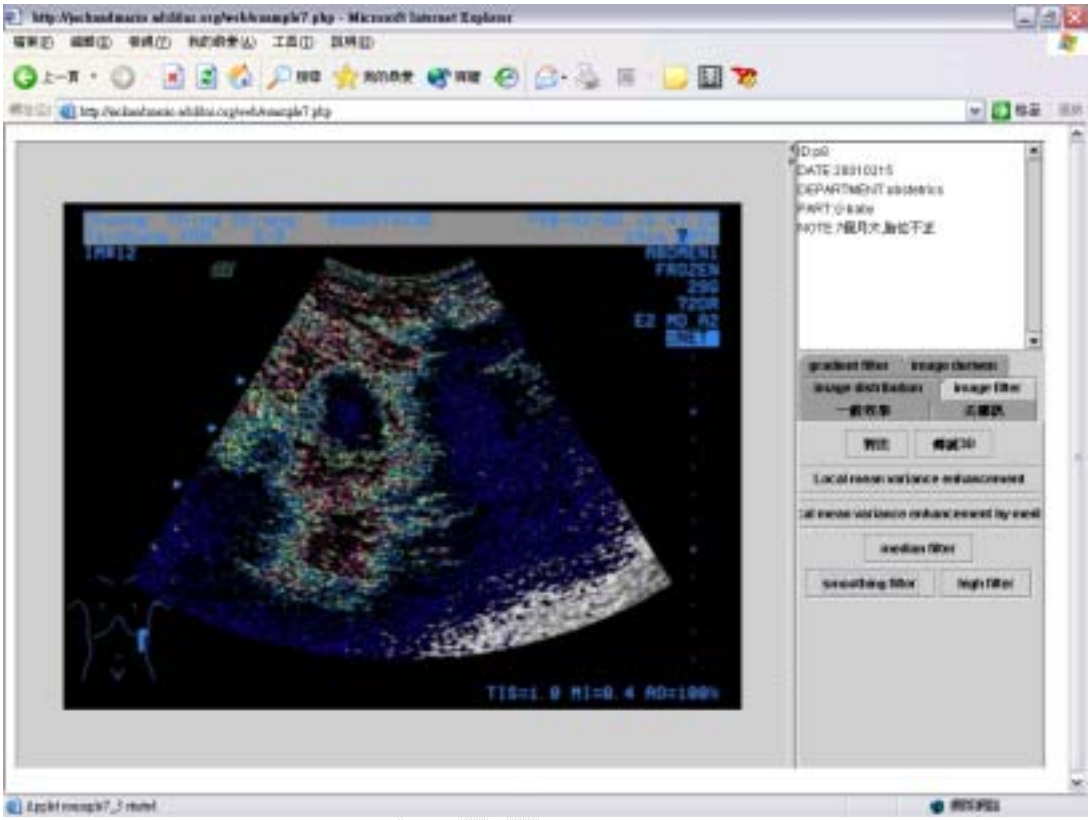
圖五十六



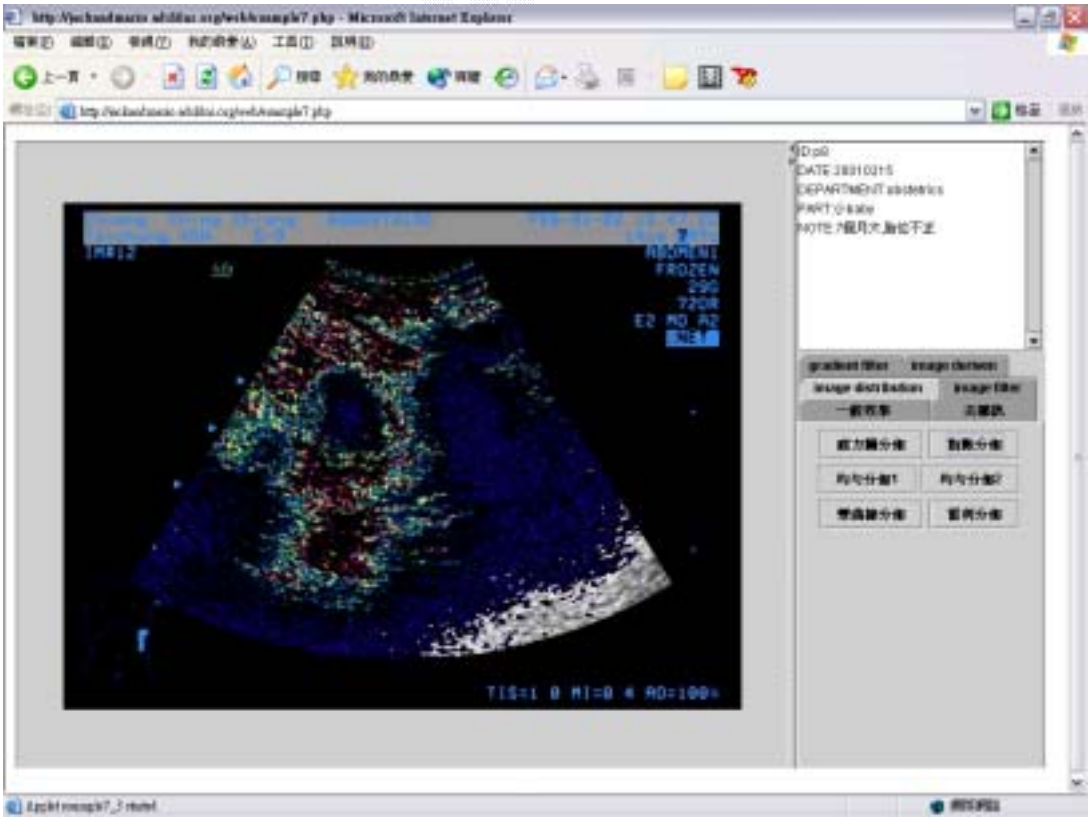
圖五十七



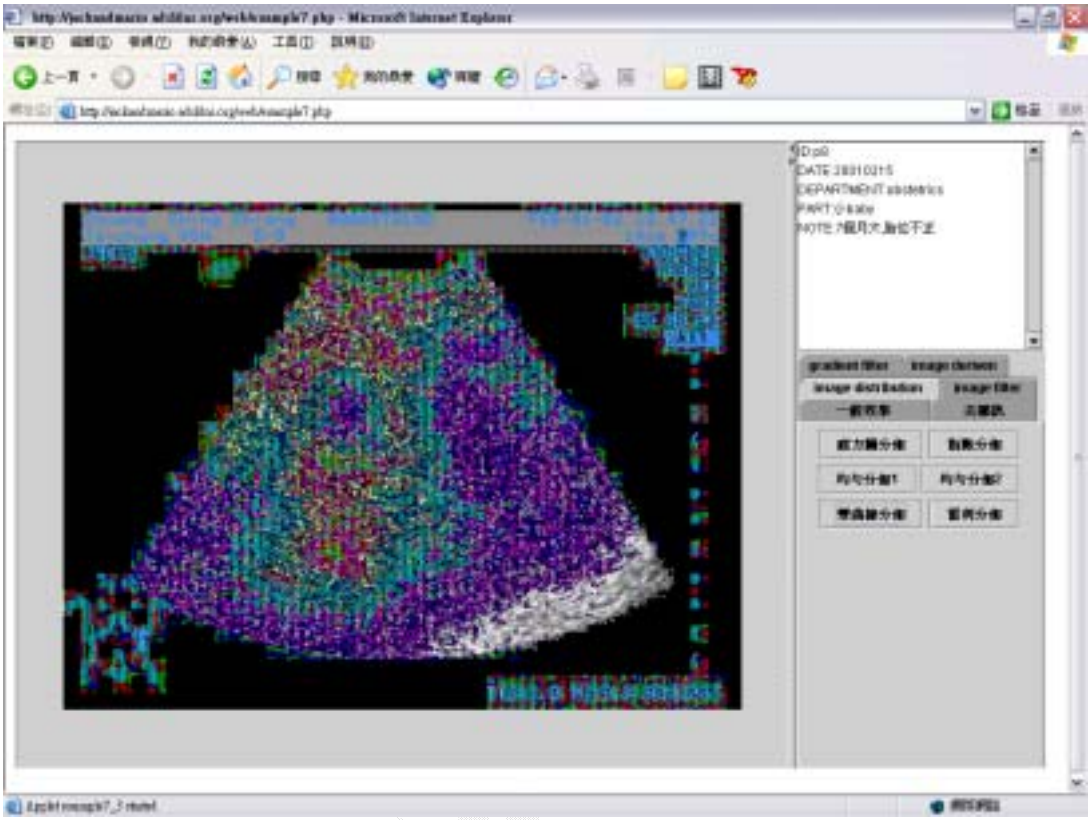
圖五十八



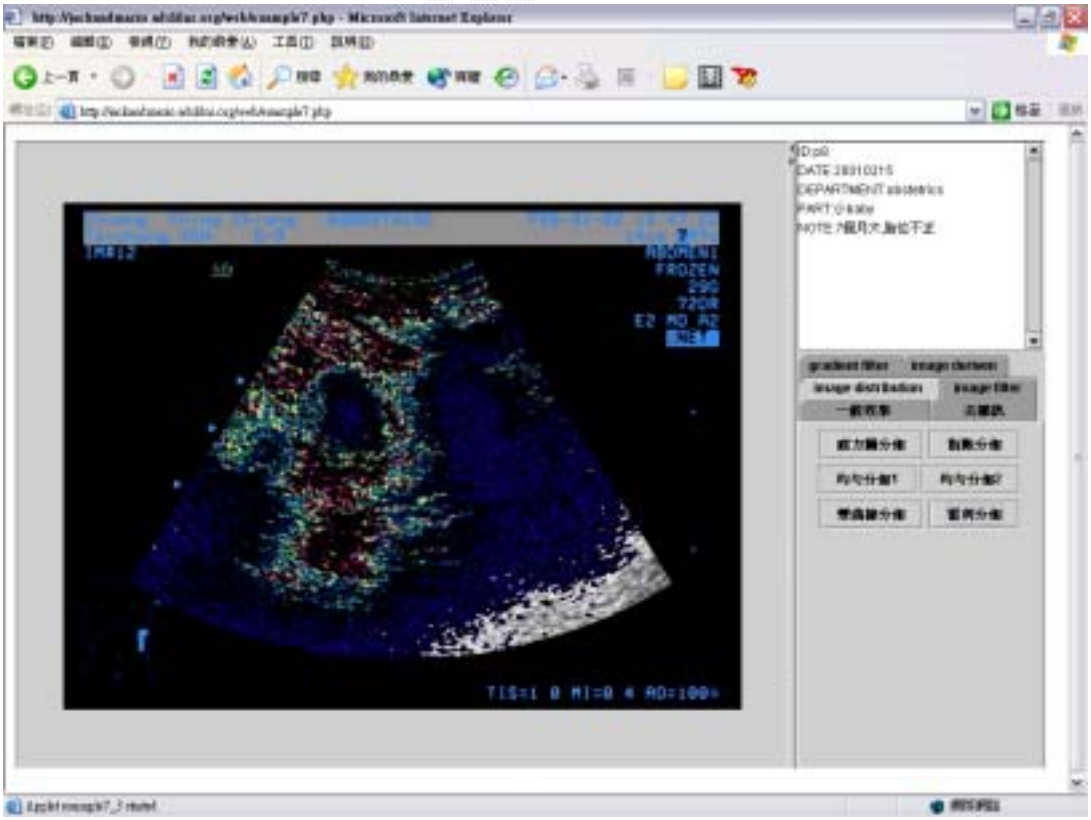
圖五十九



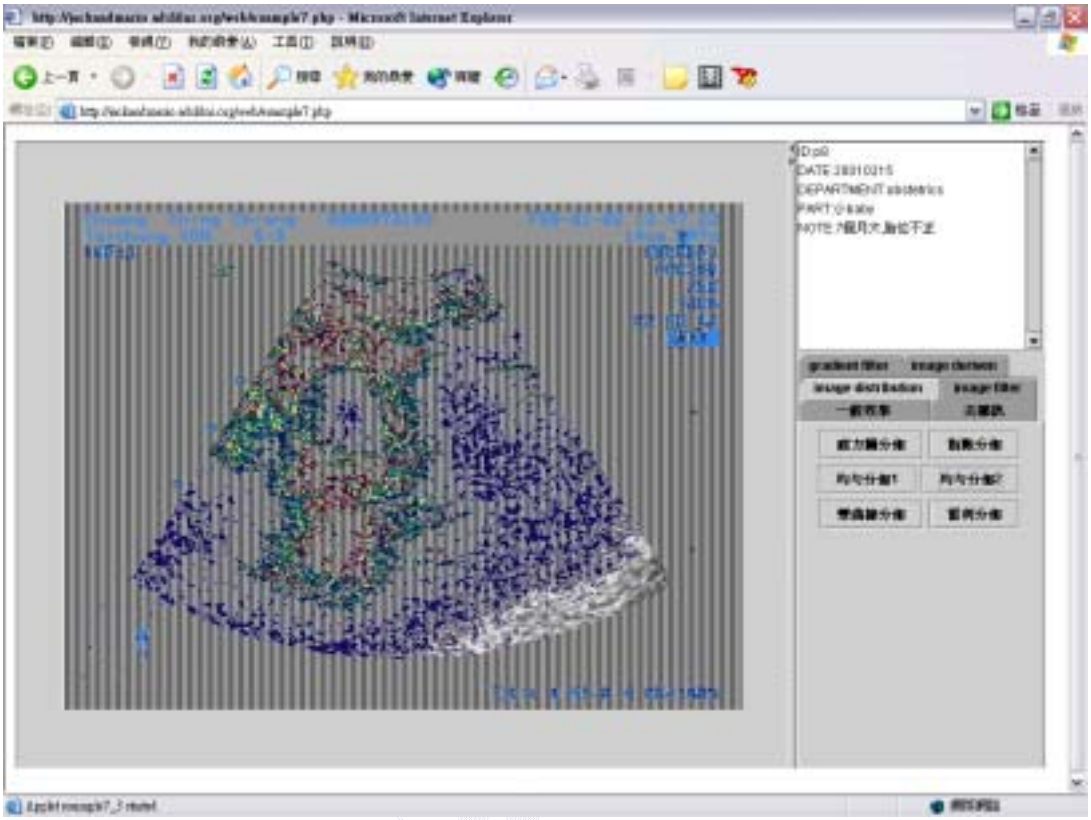
圖六十



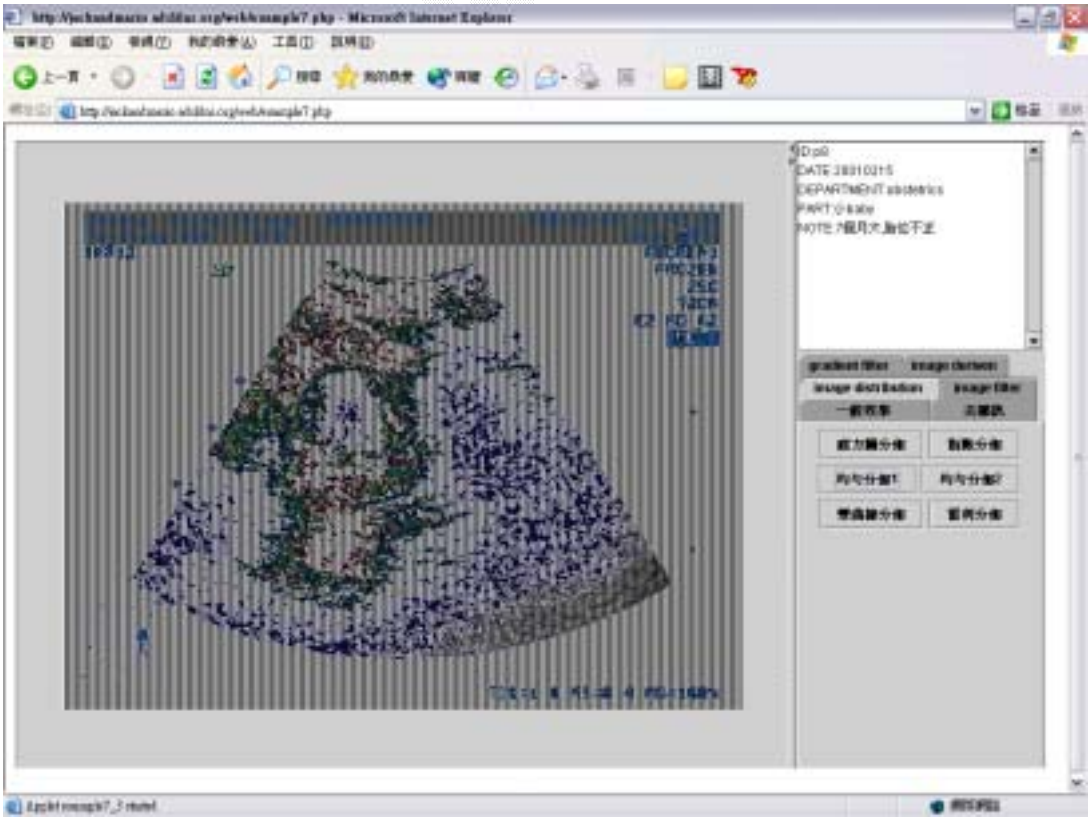
圖六十一



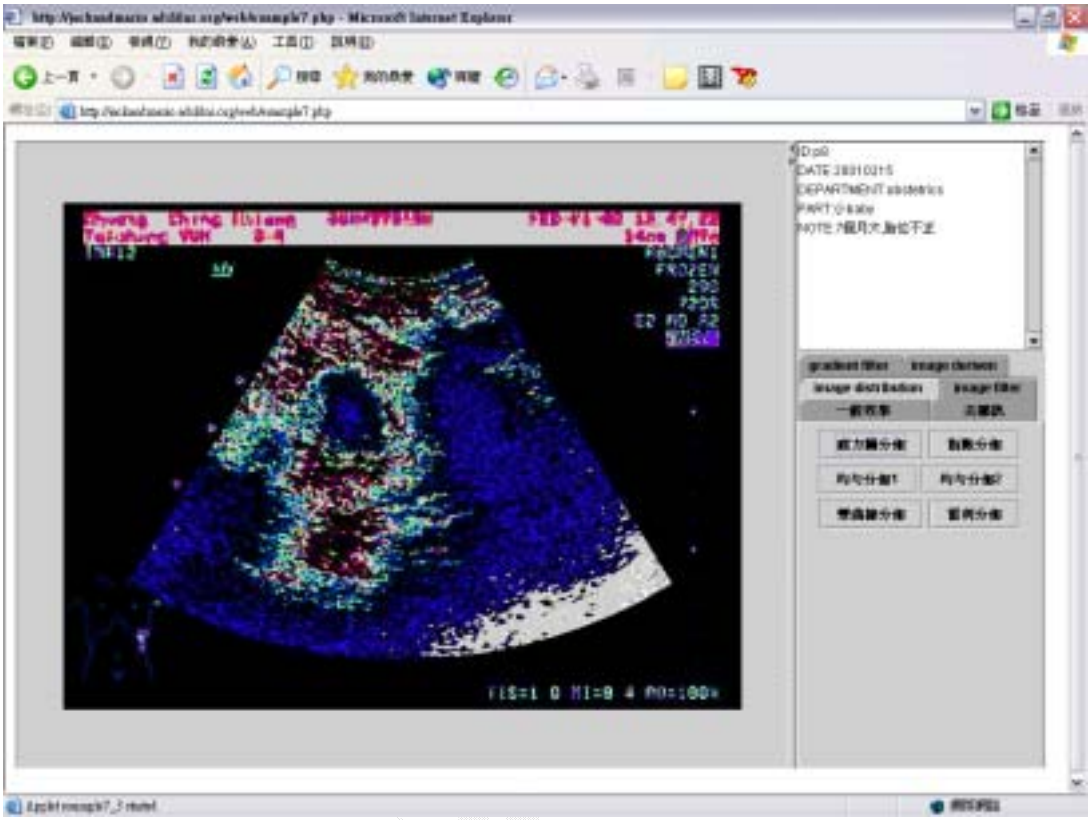
圖六十二



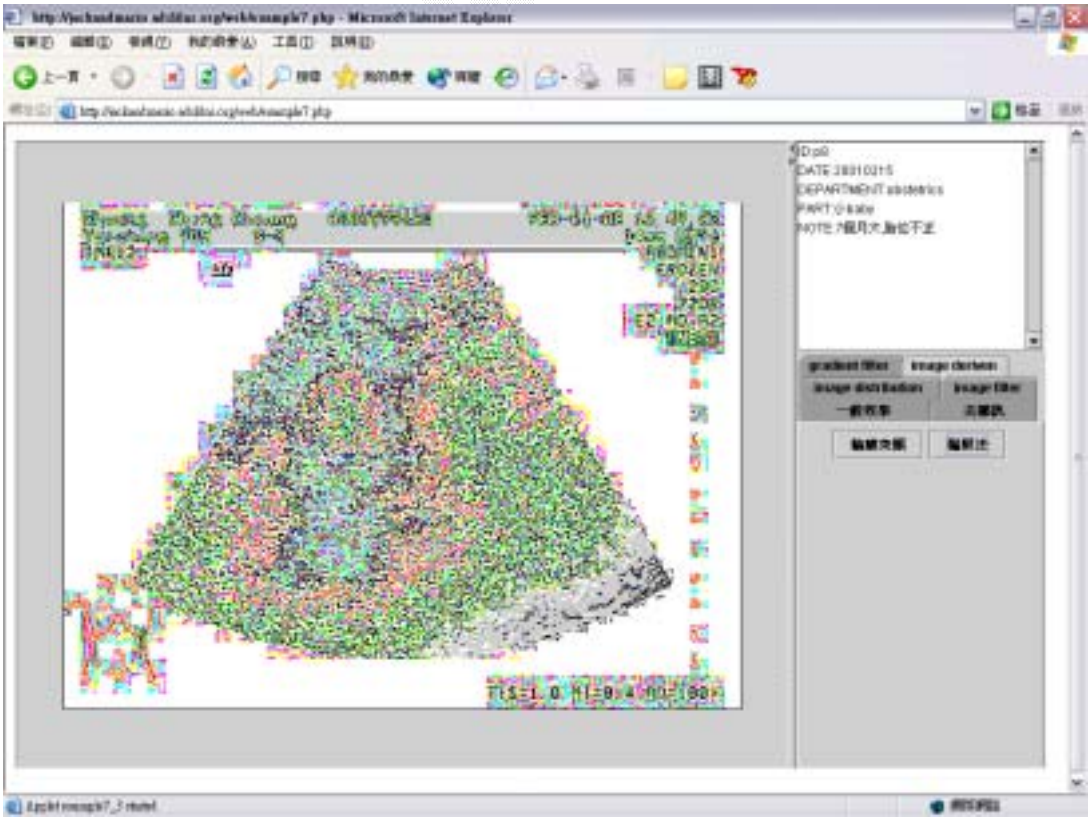
圖六十三



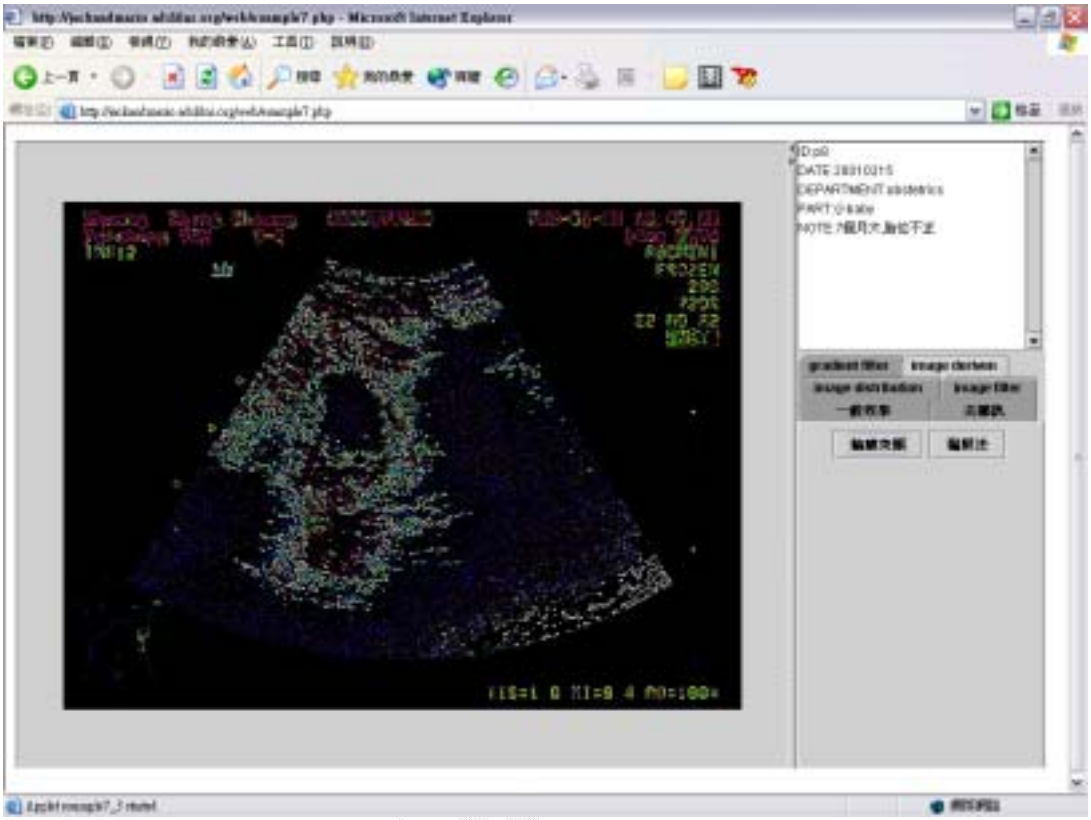
圖六十四



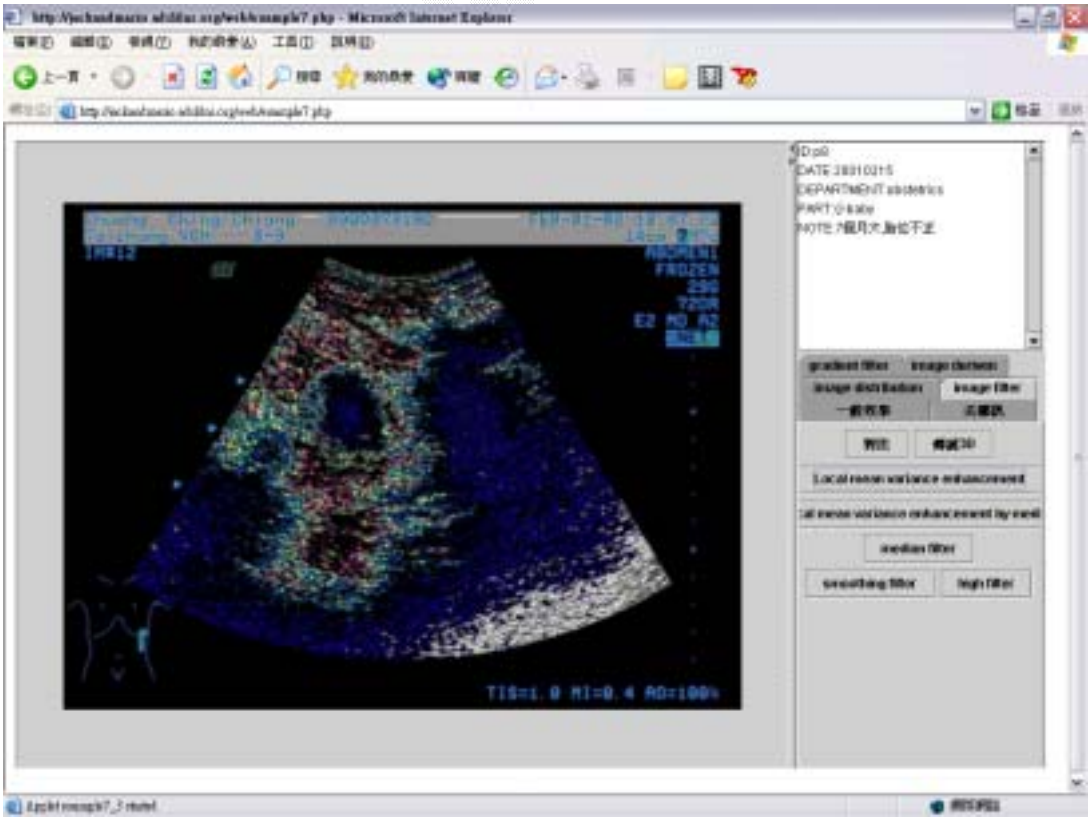
圖六十五



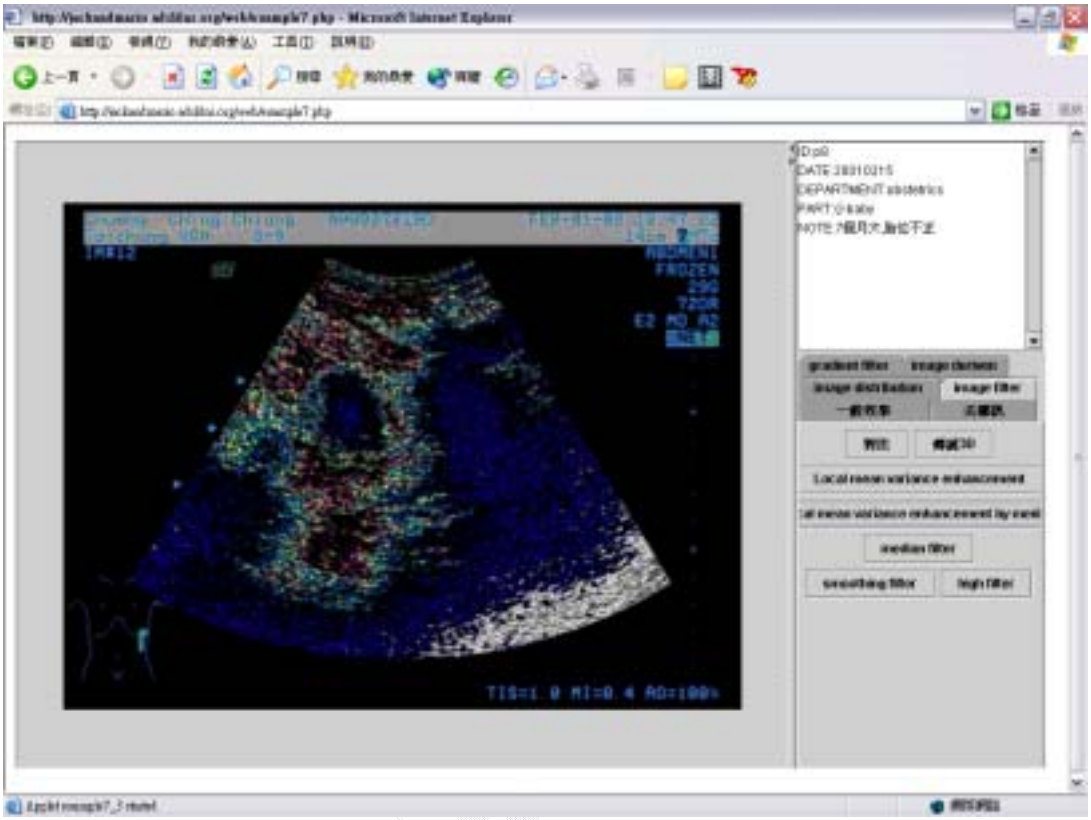
圖六十六



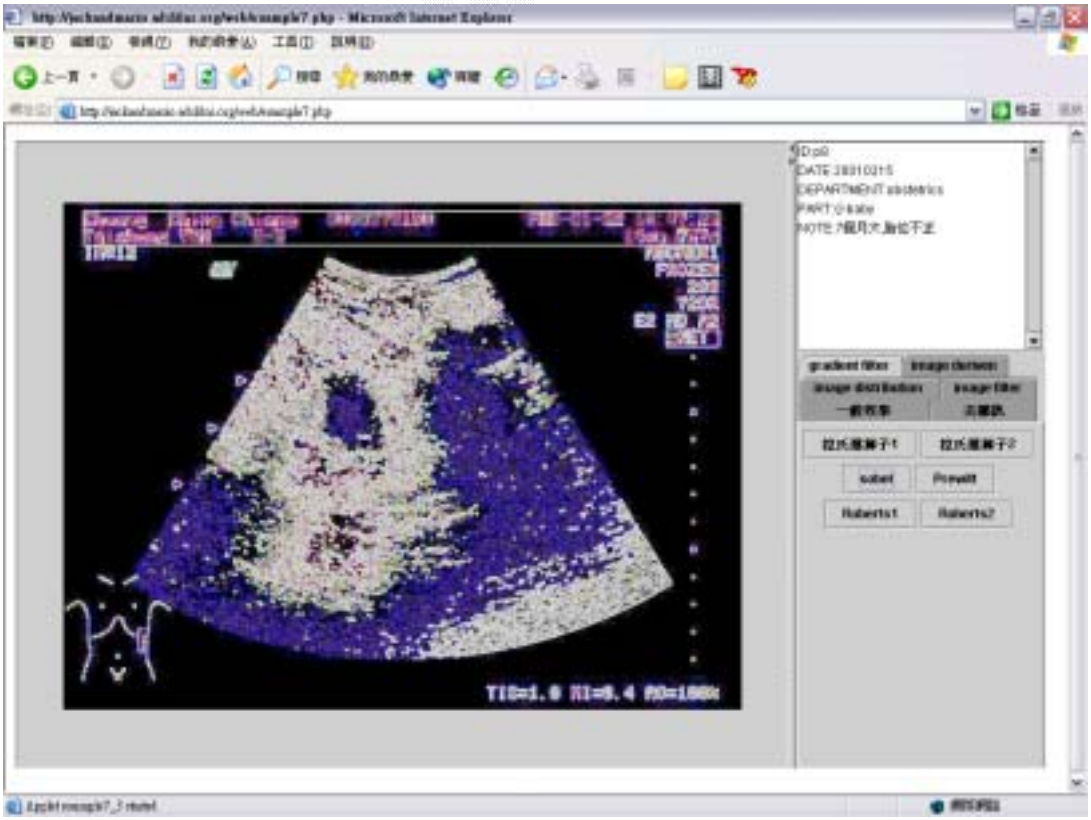
圖六十七



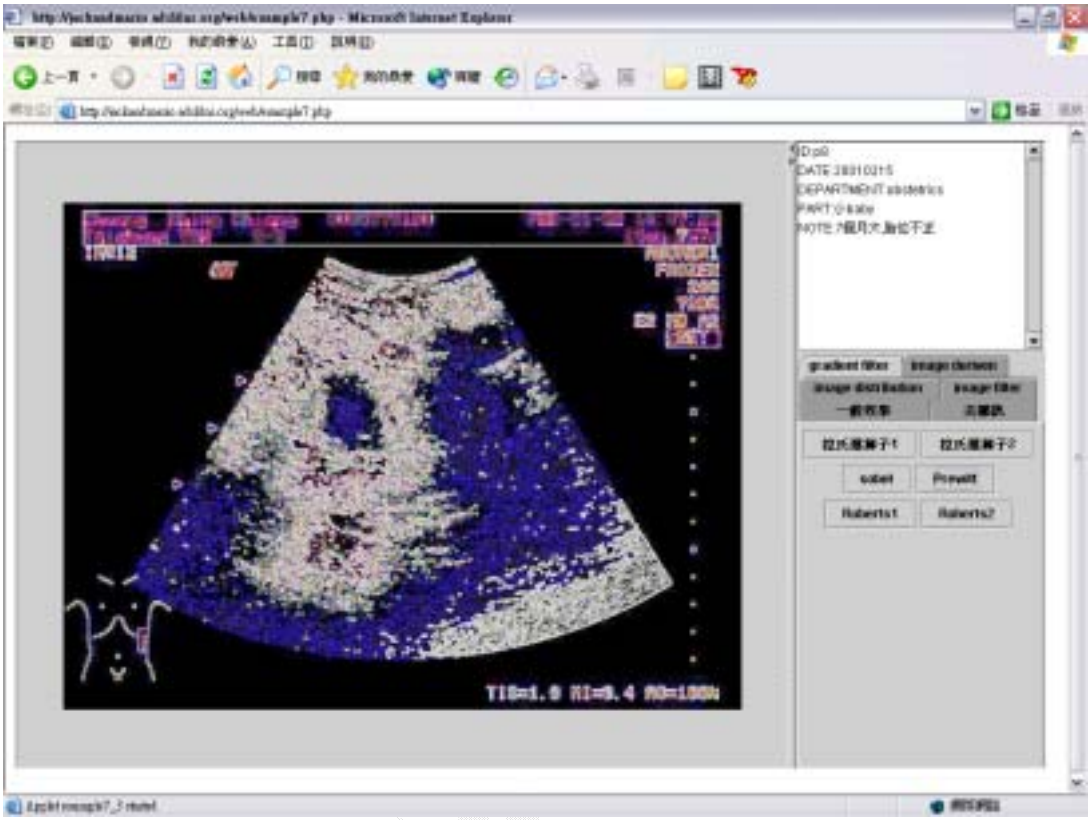
圖六十八



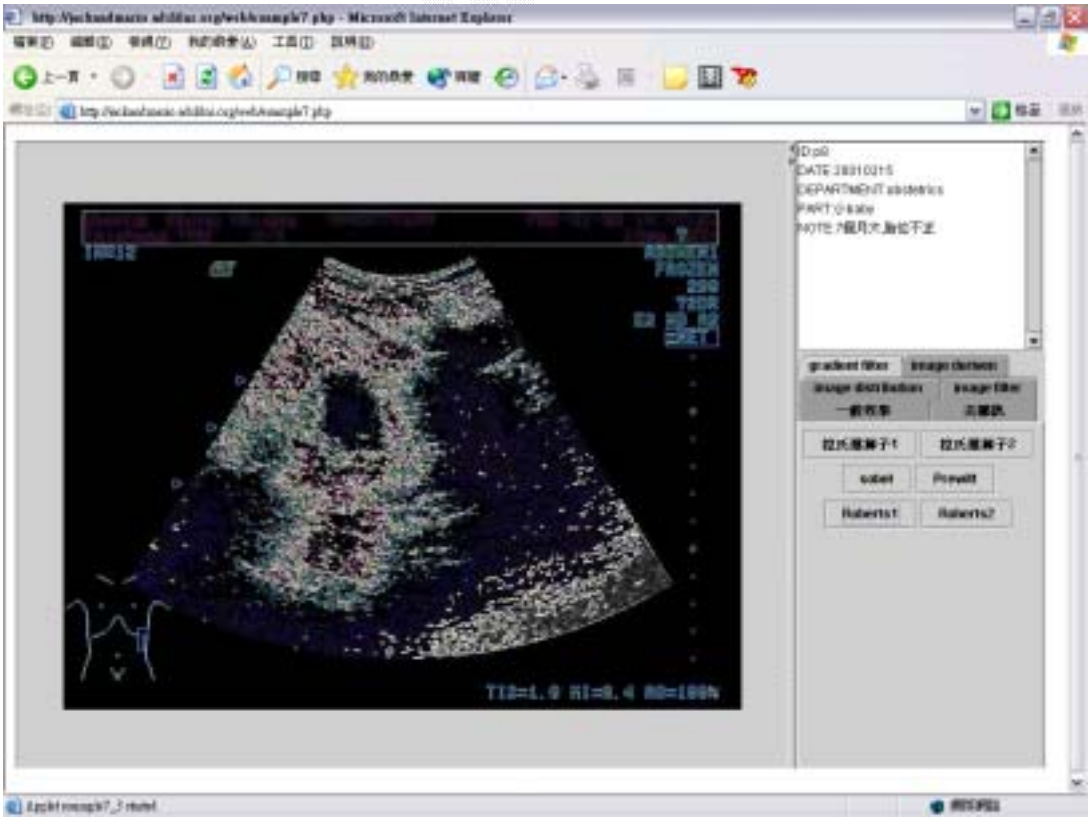
圖六十九



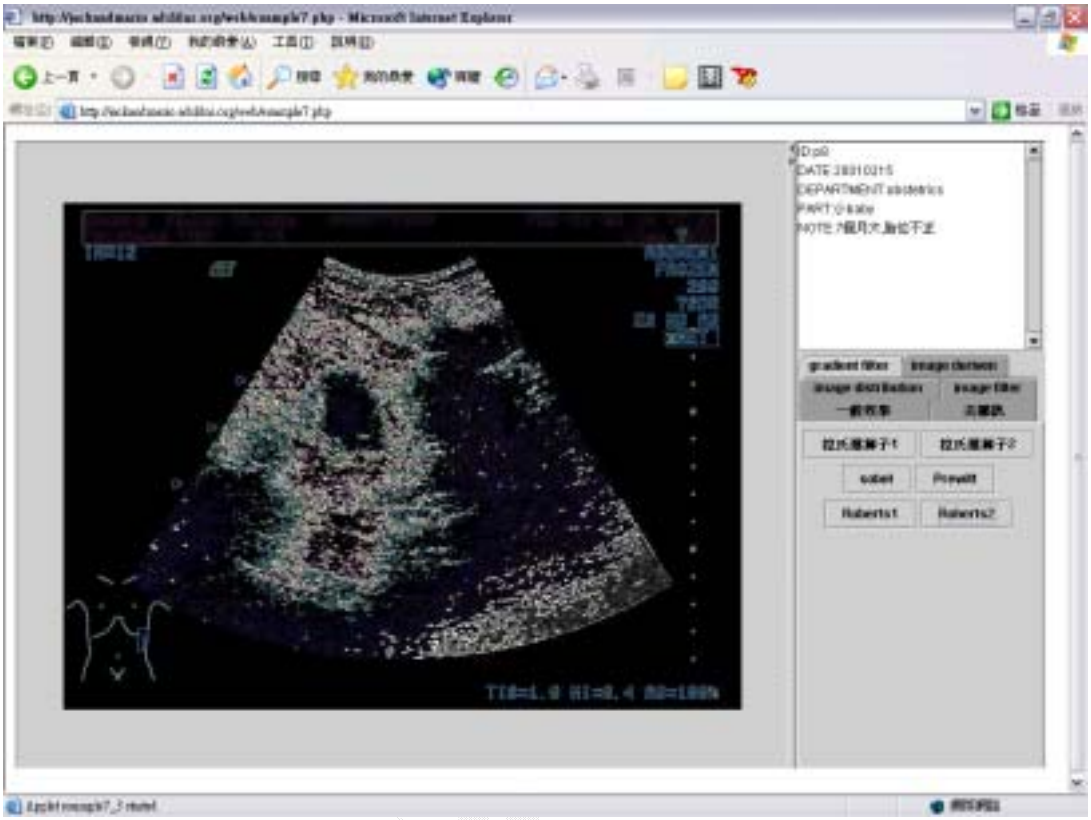
圖七十



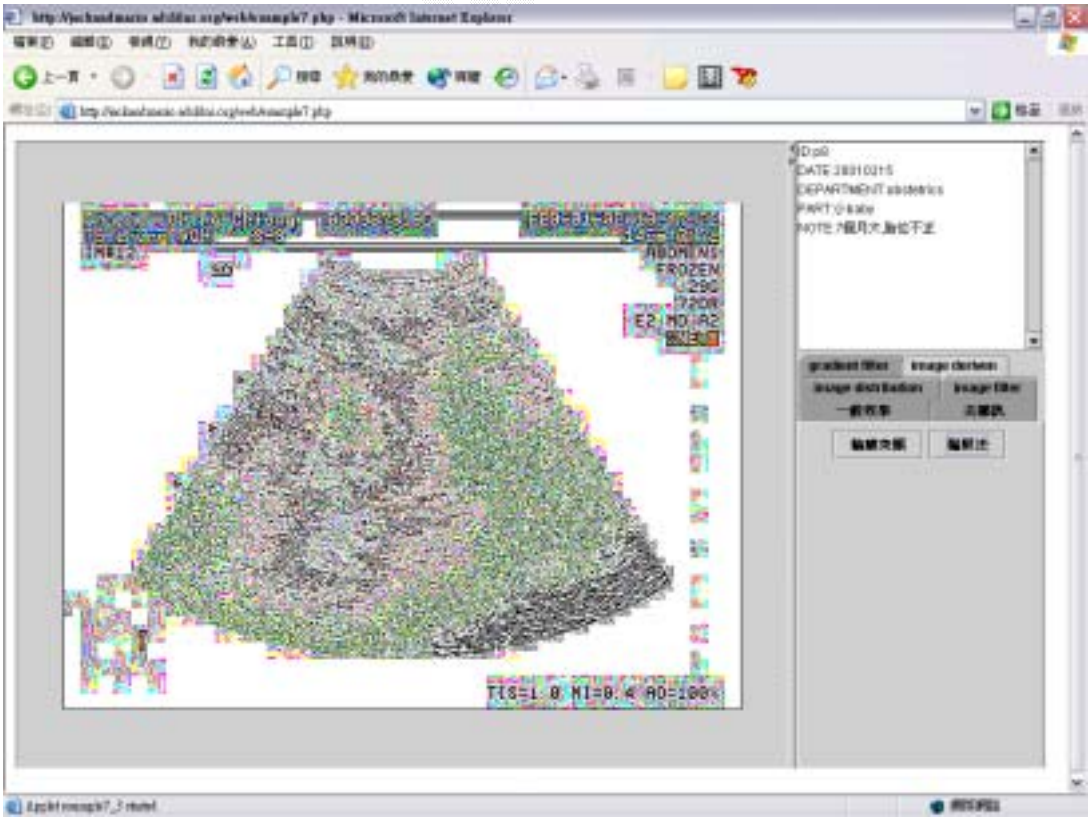
圖七十一



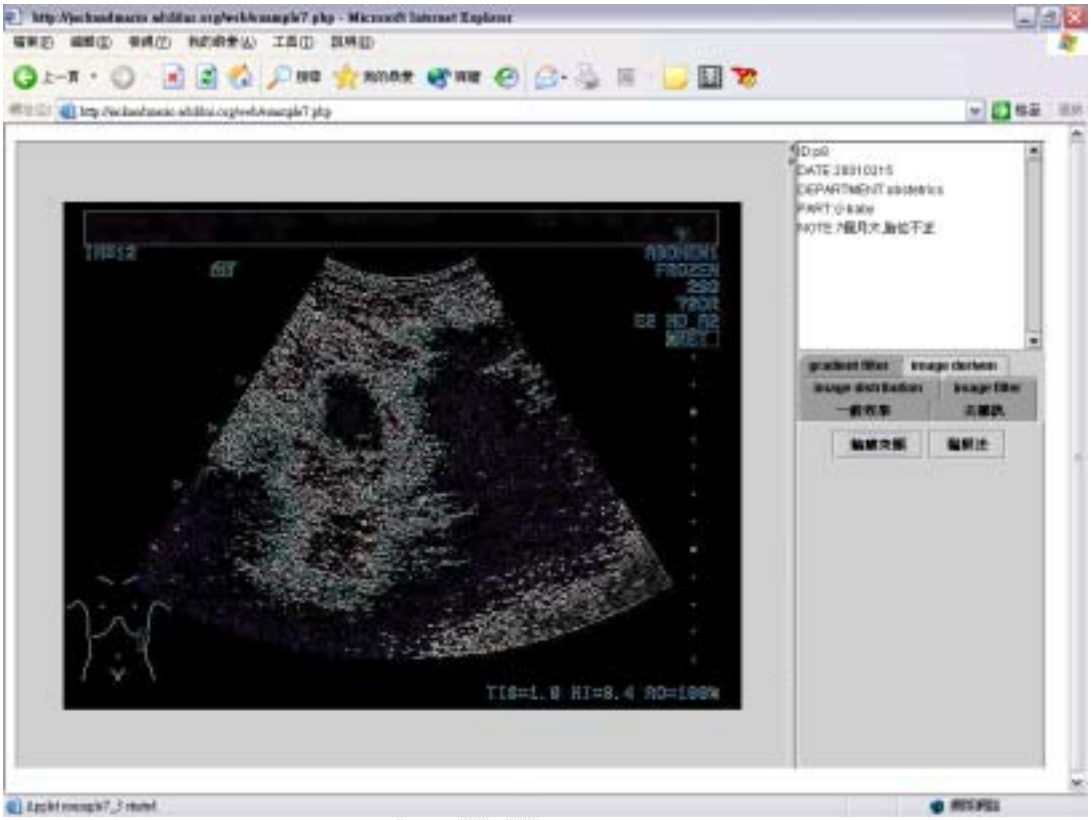
圖七十二



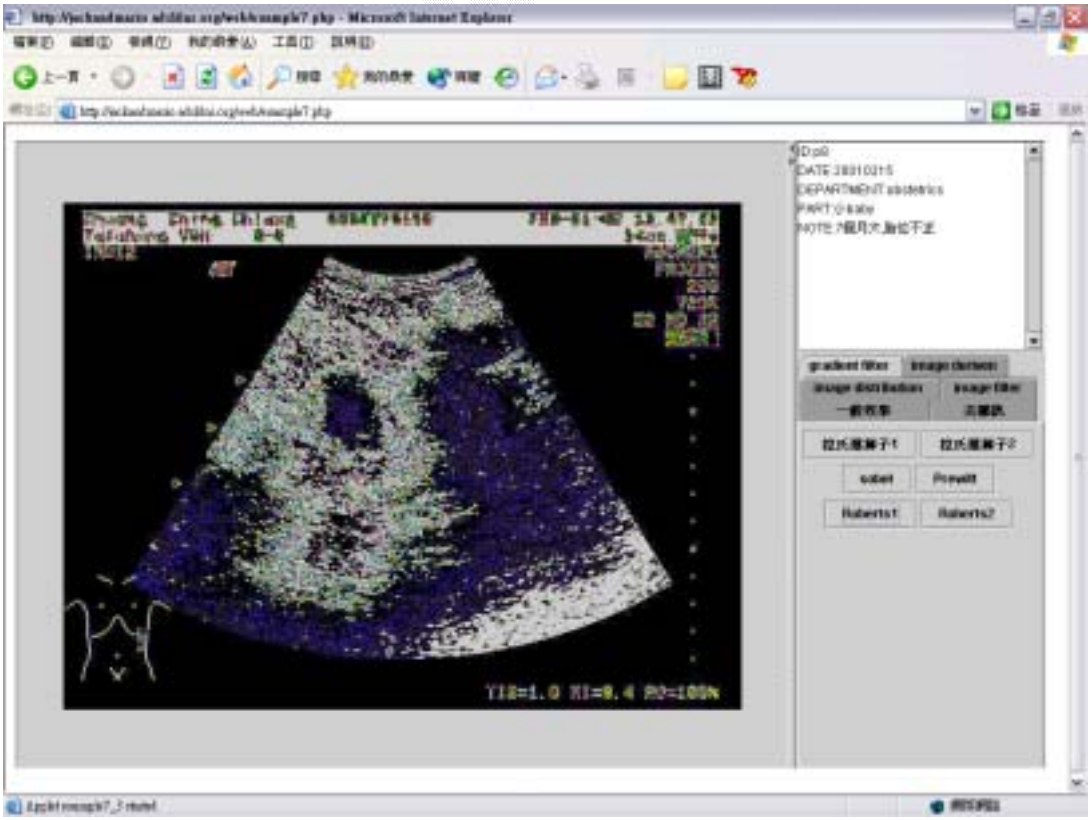
圖七十三



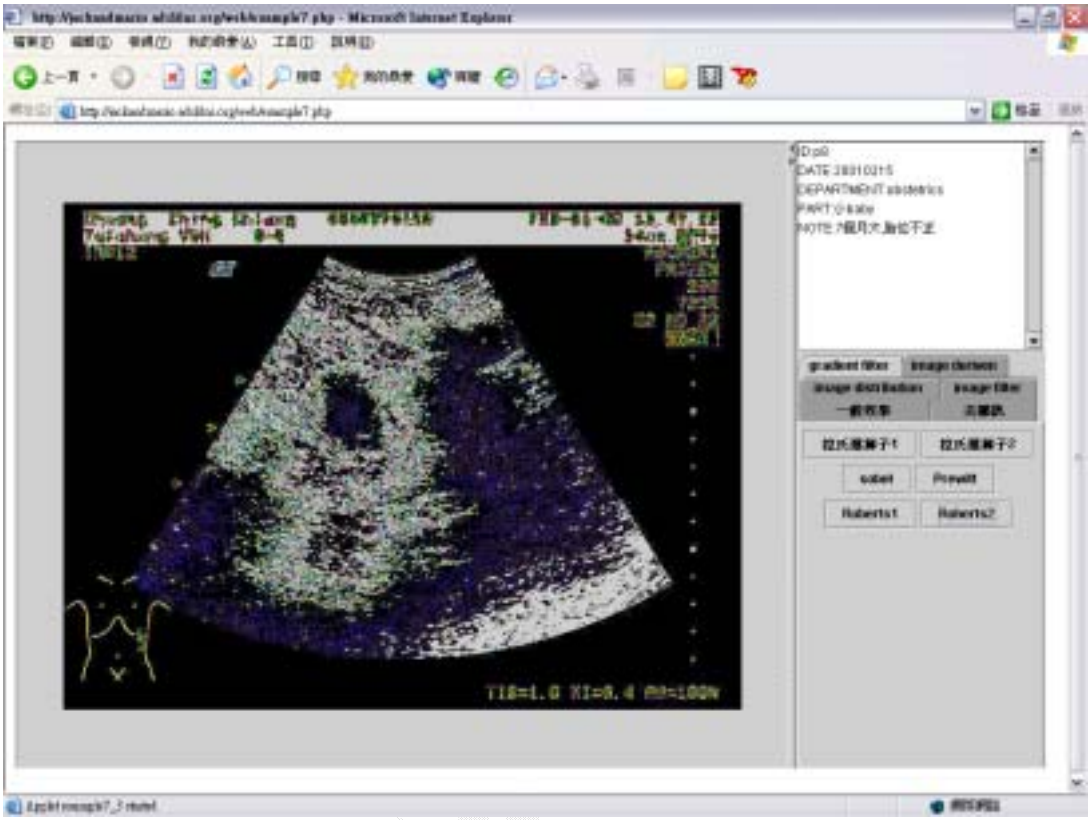
圖七十四



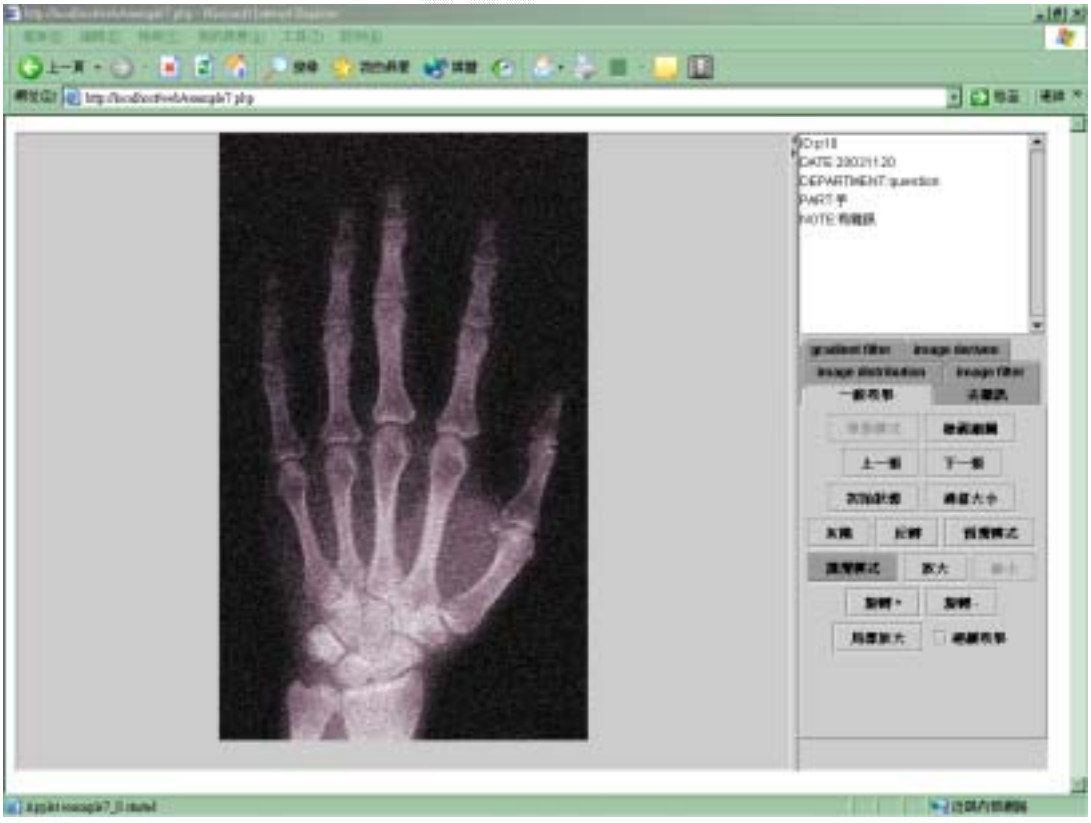
圖七十五



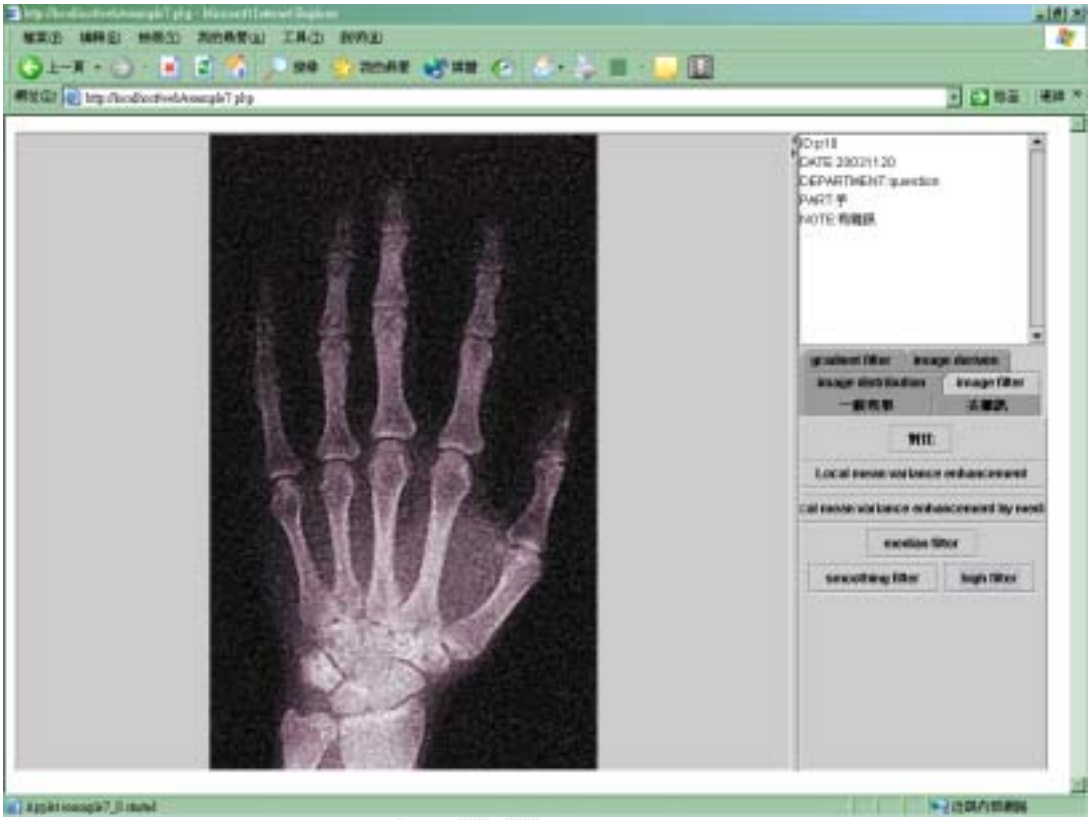
圖七十六



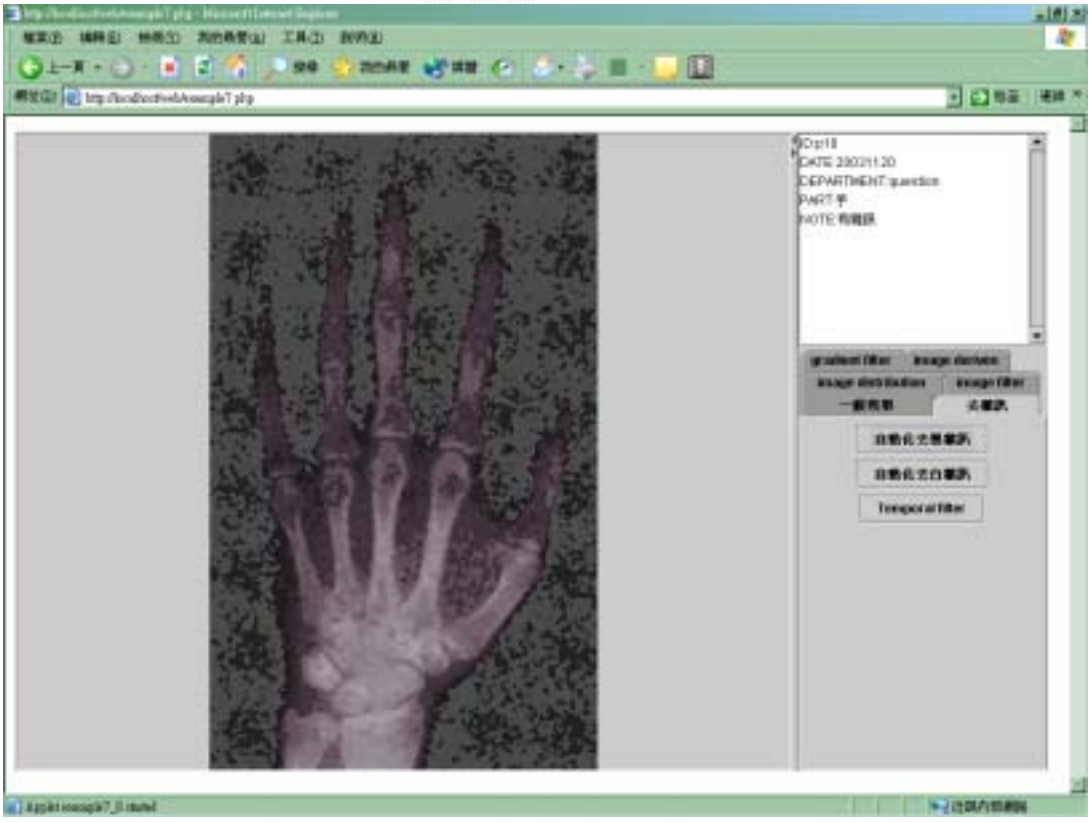
圖七十七



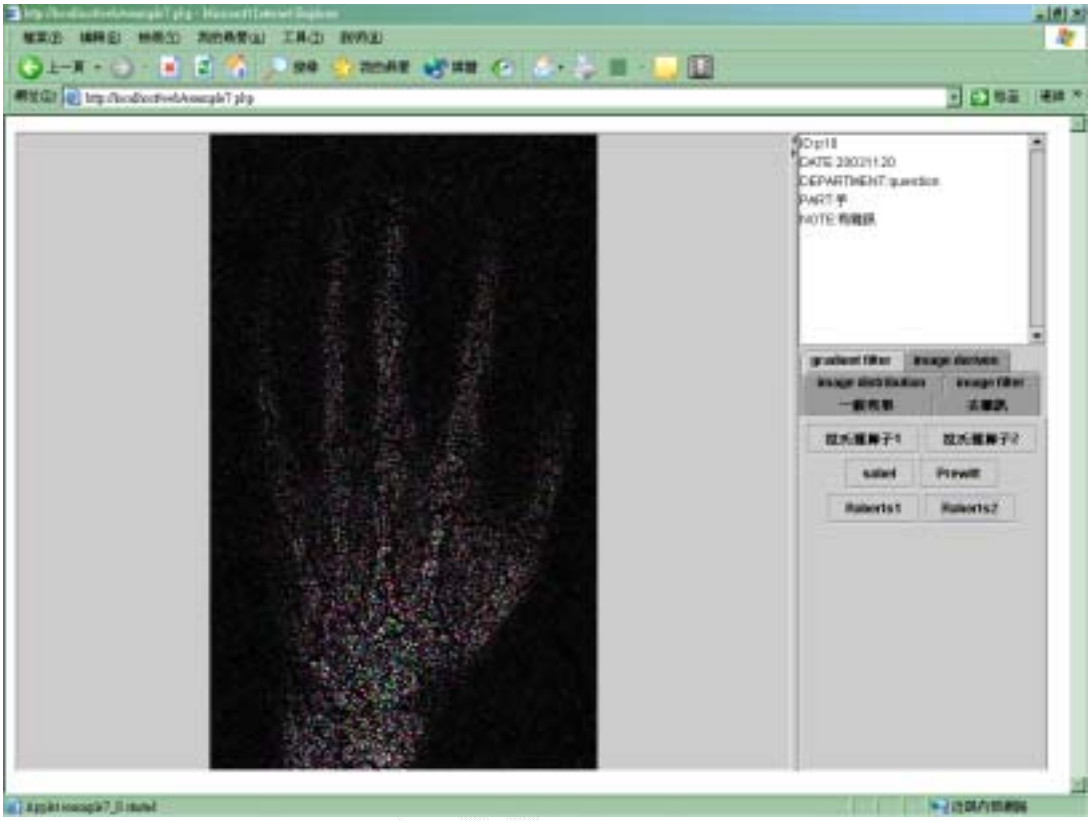
圖七十八



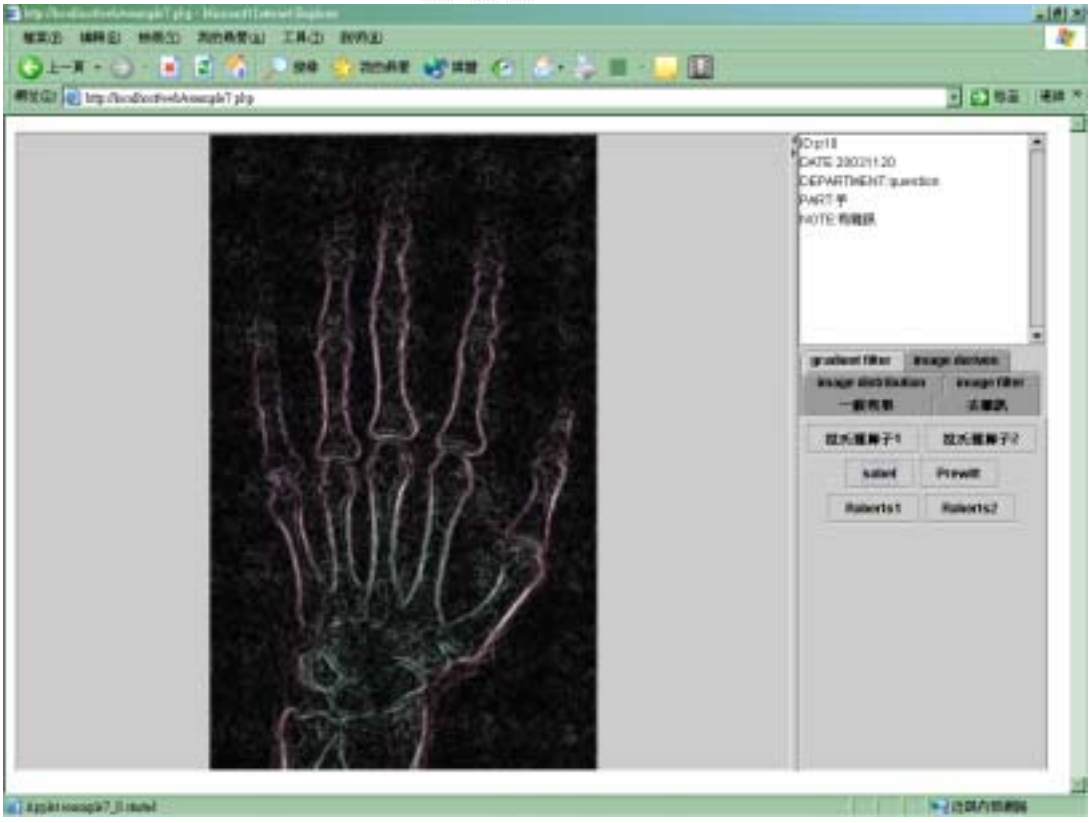
圖七十九



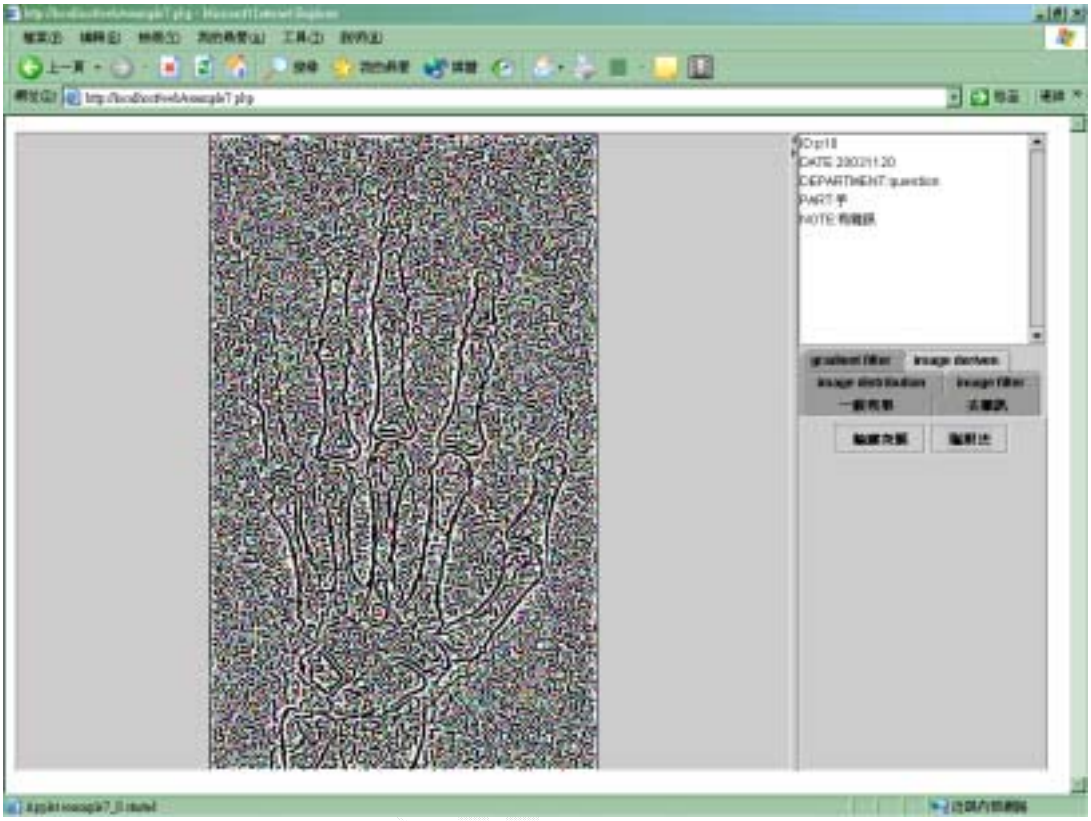
圖八十



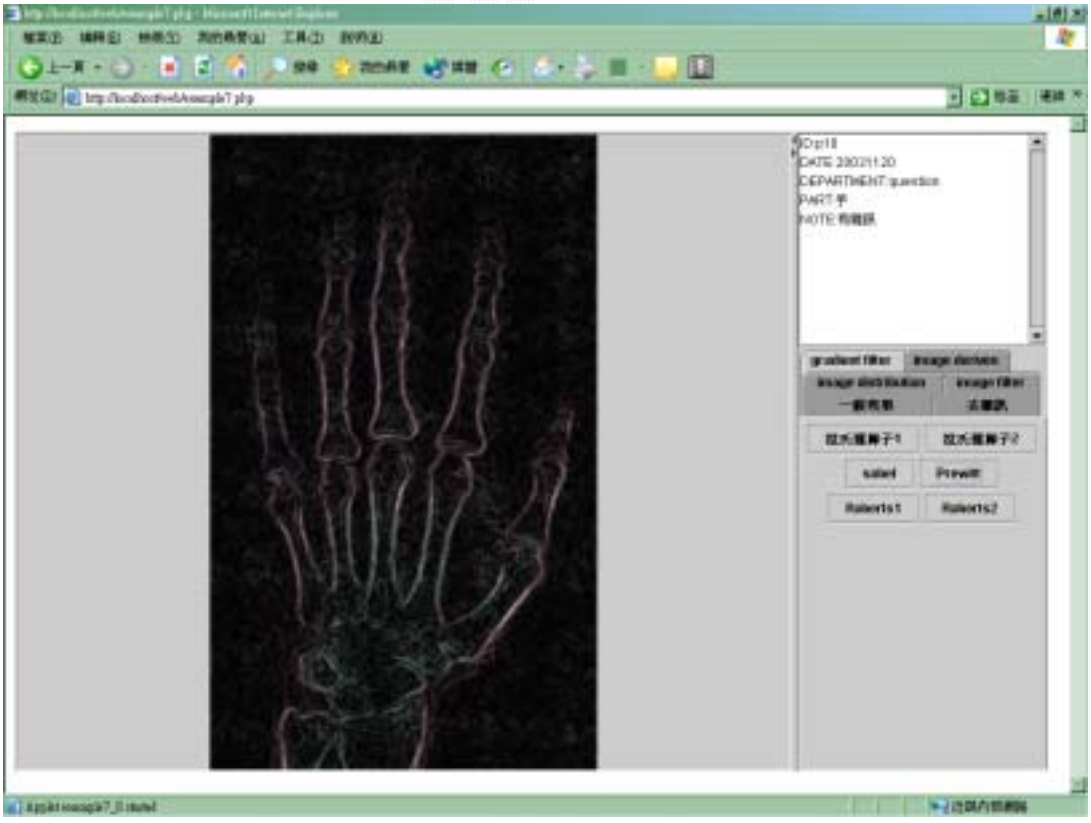
圖八十一



圖八十二



圖八十三



圖八十四

第七章　系統的改進及心得

七-1　　系統的改進

我們目前的方向基本上在提供醫療院所 e 化 X 光片、超音波圖像的實做，但還有一些功能需加強：1.介面上的更新 2.功能上的增加 3.功能上的改進

在介面的更新上，我們需要致力於人性化，讓使用者能馬上上手，且在介面上的顯示圖像區面積增大，能讓使用者在觀覽時一目了然。

在功能的增強上，我們應再加上一些功能，例如浮水印、細線化、特徵選取的功能；可以結合 VRML，使影像 3D 化，那這樣會讓此系統更加完善、功能更齊全。

在功能的改進上，我們分 4 項來改進：

第一項：功能上的加速 要再檢討功能的演算法，使畫面的結果能更快的呈現，減少使用者時間的等待。

第二項：網路上的加速 由跟主系統邏輯上分開的子系統進行壓縮傳送，所以此子系統在傳送小瀏覽圖的效率大大地影響整個系統的速度；因目前傳的小瀏覽圖未經過壓縮處理，如果能經此處理，一定能增加不少客戶端畫面的顯示速率跟減少網路的負擔，不過壓縮不在我們專題討論的範圍內，所以等有餘力在去進行。

第三項 伺服器上的加速 目前壓縮子程式跟主 SERVER(網路伺服器）在同一主機上，如果將二者分開，另開一主機提供壓圖程序，將減少網路伺服器負擔，自然能提升伺服器的整體效率。

第四項 資料庫的改進 目前的資料庫乃自行模擬醫院 X 光片的情形，因系統上的處理功能不一定只侷限於醫院 X 光片，只要應需求更改資料庫，將可實作在許多地方如圖書館或美術館。

七-2　　心得

顏嘉俊 心得：

在這個專題中，我負責的部份是”影像處理”效果的部份。”影像處理”在大學時，因為沒開該門課，所以等於是整個全部重頭讀起。還記得不懂 pixel 值被老師罵，也記得因為不了解影像的基本三原色及影像模型....等，在圖書館找出書...等經驗，真是覺得好特別。

原來想要嘗試將頻率域的方法做進入這些效果，但時在迫於時間的限制及自我的能力有限，使得本專題只有做頻慮域的一些方法。

不過境過這次的專題，我覺得我真的學了好多，也對伊些影像處理有了基本的概念，同時也感嘆電腦影像世界的浩瀚。

最後，我要感謝黃秋煌老師對我們的指導，還有林申宜學為我們解惑，我們才有機會完成這個專題。

黃耀田 心得：

這次的專題實習，雖然有做出一點東西，可是我覺得我還是太怠惰了，從以前的習慣就要人在後面逼，以致於有一段時間的空窗期，導致日後時間的緊迫。

最大的因素就是自己的抗壓性不足，每每遇到困難就停下來，讓以後的日子來彌補，但只是駝鳥心態跟增加組員的負擔。還有最重要的是缺乏主見，有自己的想法才有創造力，不過黃老師還是很熱心幫我們想題目，還介紹林申宜學長指導我們，解決我們不少的困難。

對於專題的進行，一定要想好明確的方向，才能減少人力時間。大三上學的資料庫沒學好，且設計資料庫經驗的不足，讓資料庫設計出來跟原先的有段差距，且失去了正規化條件，且在急於進行跟 Applet 整合，讓資料庫的更改成了一個麻煩工程而先放棄，這些經驗都是很好的收穫。

最後，感謝黃秋煌老師的指導，還有林申宜學長百忙中為我們解惑，指引我們方向，此專題才有機會完成。

劉奕廷 心得

專題跟平常作業有很大的不同是小組討論,像專題這樣的大題目事先的規劃就蠻重要的,在還沒正式進入 coding 之前,都要常常的討論,像這次專題,我們就到彰化醫院去實地看了有關醫師在看診方面的相關事情,了解我們的專題在醫院中擔任什麼樣的角色,然後再將工作切細,雖然我們在分工方面並沒有做很好,其實有些工作也很難分開,像 Applet 部分分成影像處理和整體顯示框架的事件處理,影像處理可以分成很多人做,但是整體顯示框架的事件處理就只能一個人擔任,這樣就會很辛苦,還好在整合上沒有大問題。

像這樣的大系統,很多人應該都會想到要系統分析,但對於初次接觸的我們就非常困難了,只能從最後的成果中,反過來拼湊出大概的模樣,不過經過這樣的過程,也粗略的知道系統分析在做什麼,有初步的印象。

最後,謝謝黃秋煌老師和林申宜學長給我們的指導。

陳志玄 心得

主要負責瀏覽介面處理,對於 Java 語言的熟悉、swing 元件的使用,事件回應的處理、applet 的應用及限制都有進一步的了解,尤其是介面的使用,必需讓使用者容易上手,及版面的配置等都是值得去探討改進的地方。

第一次開發視窗環境,對於事件回應的處理,沒有太多的經驗。例如捲軸的處理、配合旋轉、放大,所造成影像顯示位置的計算,或是滑鼠移動拖曳而造成影像的捲動等,這些都是慢慢去學習得來的。此外團隊的合作分工,彼此時間的配合及討論,這些都是十分保貴的經驗,這是自己一個人做專題所無法獲得的。當然也形成不少時間上的浪費,來完成整合、進度的掌控,這些可做日後的借鏡。

最後,感謝黃秋煌老師的指導,還有林申宜學長百忙中為我們解惑,指引我們方向,此專題才有機會完成。

參考資料

[1] G. B.Shelly and T. J.Cashman and H. J.Rosenblatt, Systen Analysis and Design Forth Edition, pp.4.2–4.19, Course Technology, 2001

[2] 吳成柯、戴善榮、程湘君、雲立實 譯(R. C. Gonzalez and Richard E. Wood著), 數位影像處理, 儒林圖書, Nov. 2001

[3] 張裕益、劉春成 譯(S. Holzner 著), JAVA 2 徹底研究, 博碩文化, Nov. 2002

[4] 張晏誠 譯(J. Knudsen 著), JAVA 2D 圖學技術, O'REILLY, March 2001

[5] 葉涼川譯(Ladd 著), Java 演算法, 麥格羅.希爾

[6] 王遠帆譯(E. Tonny 著), Java 殺手級程式,松格出版社

[7] 陳同孝、張真誠、黃國峰著, Digital Image process 數位影像處理技術, 旗標出版股份公司

[8] 陳同孝、張真誠、黃國峰著, Electronic Imaging Techniques 電子影像技術, 旗標出版股份公司

[9] 袁葆宏 劉豐豪 吳建和著, JAVA2 語言實務, 旗標出版, May 2001

[10] 蕭進松著, 數位影像處理, 全華科技圖書, Sep. 2001

[11] 連國珍,數位影像處理,儒林圖書,Oct. 2002