國立交通大學

資訊科學與工程研究所

碩 士 論 文

區塊式濾波反投影演算法設計與 FPGA 實作 (I)

Block-based Filtered Backprojection Algorithm Design and Implementation on FPGA (I)

研 究 生：吳俊瑋

指導教授：荊宇泰　教授

中 華 民 國　九 十 五　年 九 月

區塊式濾波反投影演算法設計與 FPGA 實作(I)

Block-based Filtered Backprojection Algorithm Design and Implementation on FPGA (I)

研 究 生：吳俊瑋　　Student：Chun-Wei Wu

指導教授：荊宇泰　　Advisor：Yu-Tai Ching

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

June 2006

Hsinchu, Taiwan, Republic of China

中華民國九十五年九月

區塊式濾波反投影演算法設計與 FPGA 實作(I)

學生：吳俊瑋　　指導教授：荊宇泰　博士

國立交通大學資訊科學與工程研究所

摘要

濾波反投影(filtered backprojection)演算法在電腦斷層掃描扮演相當重要的角色，其計算繁瑣及複雜。我們目標設計一個可高度平行化處理的演算法，由於硬體資源不足，無法對整張 filtered sinogram 做影像重建。因此採用區塊式重建，逐個區塊做影像重建，即為最後結果。經由軟體模擬，模擬結果有大小不同的區塊，我們都得到預期的結果。此論文需參照林坤世同學所撰寫的”區塊式濾波反投影演算法設計與 FPGA 實作(II)”。

Block-based Filtered Backprojection Algorithm Design and Implementation on FPGA (I)

Student：Chun-Wei Wu　　Advisor：Dr. Yu-Tai Ching

Department of Computer and Information Science

National Chiao Tung University

Abstract

The filtered backprojection algorithm plays an important role in computational tomography, its calculations are complicated and tedious. All we want to do is to design a highly parallel algorithm to solve the

limitations of hardware resources. Because of lack of resources, we take the block-based policy, doing reconstruction block by block. After the final block has been reconstructed, the block-based FBP is done. By implementing the algorithm we design, we have the expected consequences in different block size. You should refer to the thesis “Block-based Filtered Backprojection Algorithm Design and Implementation on FPGA (II)” written by Kun-Shih Lin.

誌謝

首先我要感謝我的指導教授荊宇泰老師，這兩年多的時間不斷的鼓勵及教導，終於完成此篇論文，也感謝口試委員唐教授、謝博士，因為有你們的校正及督導，使我的論文更加完整，感謝我的家人的支持及鼓勵，還有我身邊的朋友及實驗室同學，這兩年多的陪伴及成長，讓我在碩士生涯學習到更多面對人生的態度。感謝張騰介學長，在我們論文萌芽期，給予幫助及意見。也感謝我的女友玟伶，感謝你的支持和安慰，讓我更成熟養成研究態度。最後感謝曾經幫助過我、鼓勵我的所有師長及朋友，因為有你們，才有今天的我，謝謝。

目錄

圖目錄

第一章　緒論

自從 1895 年德國物理學家 Roentgen 發現 X 射線以來，其在醫學影像領域的應用就受到世人的關注。然而，由於受到電腦技術的局限，真正的臨床應用時期直到二十世紀後半期才突顯出來。數學家 Radon 早在 1917 年和 1919 年分別提出 Radon Transformation 及 Inverse Radon Transformation 公式，表現在影像處理上意義即為後來的投影算變化(投影算子)。二十世紀 50 年代初期，美國神經外科醫生 Oldendorf 為了克服普通 X-Ray 成像組織結構重疊僞影（artifact），發表了第一篇 CT（Computer Tomography）論文，此時才稱為電腦斷層掃描技術；1963 年，南非物理學家 Cormack 為了準確估計 X-Ray 在組織間的衰減率，第一次把非迭代的級數理論引入 CT 重建演算法中；1968~1972 年，英國工程師 Hounsfield 位了區別大腦的灰質和白質，在老闆的資助下，製造了世界上第一台商用電腦斷層掃描機。Cormack 與 Hounsfield 在 1979 年獲得諾貝爾獎，並且在影像處理界，設置了 Hounsfield 獎。近 20 多年，CT 在理論演算法的研究一直處於極為活耀的地位，而且新進的成像方式，例如：MRI（Magnetic Resonance Image）及 PET（Positron Emission Tomography）等，在算法上與 CT 也很相似。

我們所使用重建的演算法，是以 Radon Transform 為基礎來做影像重建，Radon Transform 為物質密度函數沿著直線的線積分，配合上影像重建濾波器（filter），此稱為濾波反投影（Filtered Back Projection），是一種快速且有效率的影像重建演算法。隨著時代演變，電腦斷層掃描應用領域與日俱增，除了醫學影像方面，亦應用於日常生活中，例如：機場、港口安全檢查，對於旅客及行李進行危安偵測。或是在生物科技、農業觀察動植物的生長情形等。但是醫學上使用的電腦斷層掃瞄機器過於龐大，活動性不佳，以及掃描方式過於死板、造價昂貴等因素，使的無法有效普及於現今社會。

由於半導體技術與電路設計軟體的快速成長與成熟，目前的積體電路設計主流為系統單機片（System On Chip，SoC），此技術可整合多個完整的系統，將完整的系統模組

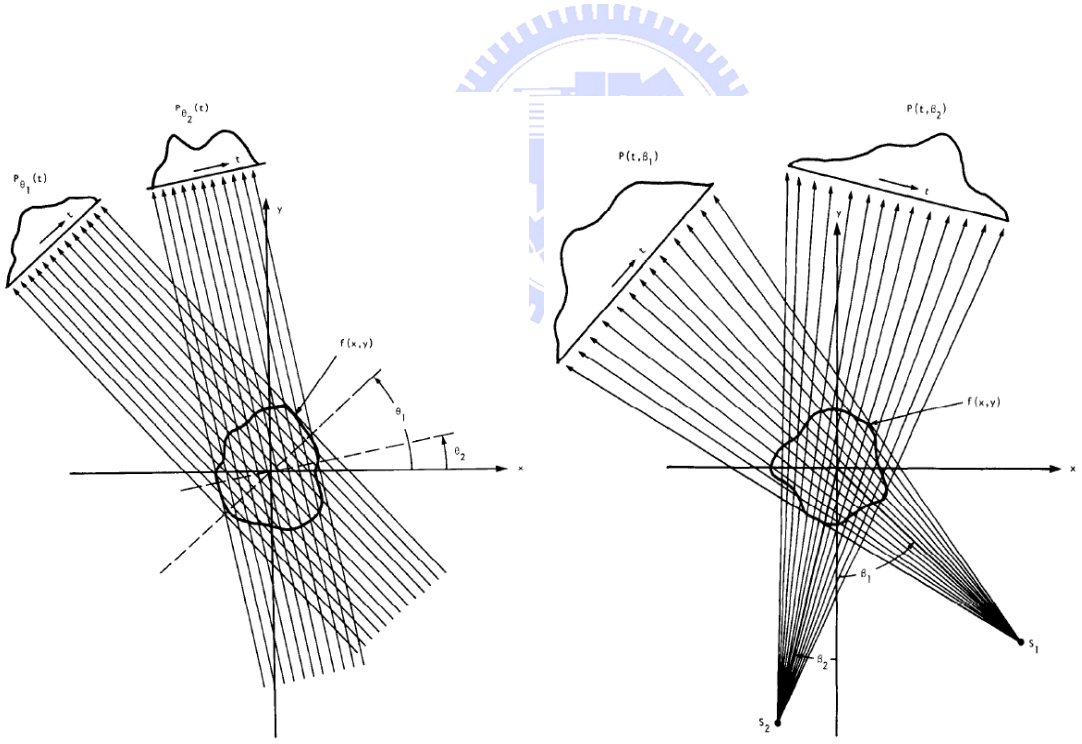
植入晶片中，不需多顆晶片即可擁有完整的功能。可減低以晶片為核心等大型機器的生產成本，亦可以縮小儀器體積。

本論文參考林坤世同學所撰寫的”區塊式濾波反投影演算法設計與實作 II”，方可更了解本演算法設計之目的。

第二章 影像重組理論

電腦斷層掃描(Computer Tomography，CT)利用 x-ray，超音波或是放射線同位素等對待測物體進行不同角度的投影(Projection)，接著在投影的反面利用一連串的探測器對透射出的 x-ray 等能量進行衰減量的偵測。這些資料我們稱為投影資料(Projection Data)。透過一些影像重建演算法，我們可以利用這些不同投影角度得到的投影資料重建出待測物體的截面圖。

對物體進行投影而收集投影資料的方式約有兩種，(1)平行投影(parallel projection)、(2)扇形投影(fan beam projection)。如圖 2-1 所示：



(a) (b)

圖 2-1 (a)平行投影，(b)扇形投影

我們的演算法是以平行投影為基礎的影像重建演算法。首先，我們會先介紹電腦斷

層攝影的基本概念，最後將介紹實際使用的演算法，濾波反投影(filtered back projection，FBP)演算法。

2.1 影像重建演算法

影像重建演算法的主要問題在於如何利用投影資料來尋找物體截面圖上各個點的線性衰減係數。例如我們以x-ray對生物組織進行投影，則我們可以在投影面的另一邊偵測到x-ray的衰減量。在這裡我們考慮物體的每個點有不同的x-ray衰減係數(物質密度不同)，而每個偵測器得到的衰減量，就是對應的x-ray射線透射物體時直線經過的點的衰減係數總和(類比物質密度總和)，相當於是x-ray射線經過物體的那條路徑的線性積分(line integral)。如圖 2-2 所示：

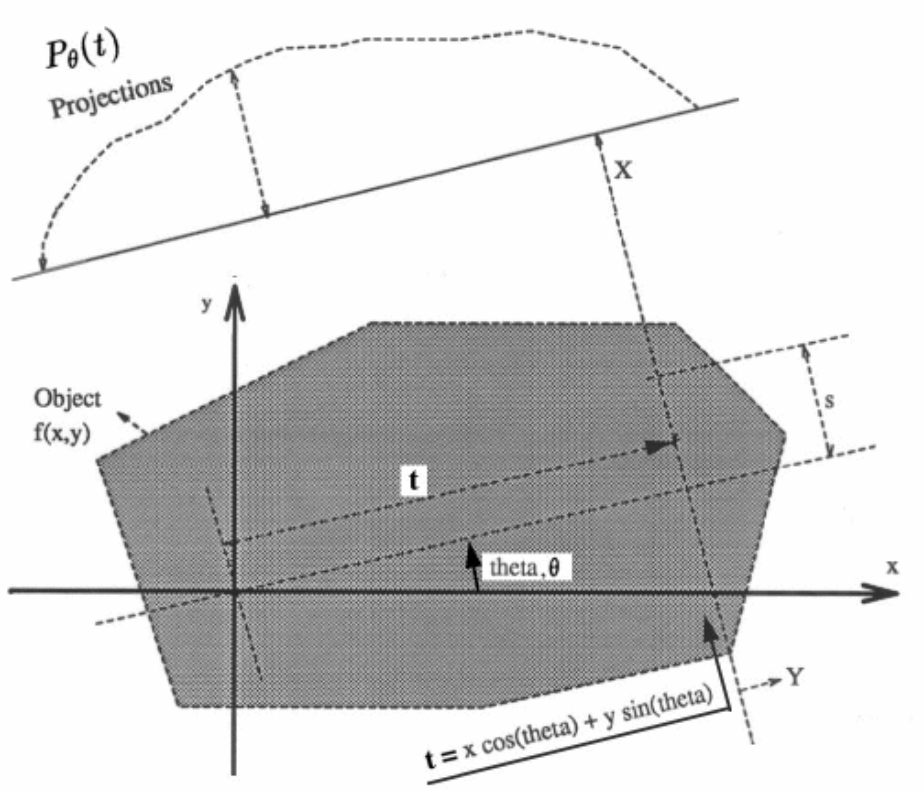


圖 2-2

由上圖可知我們以二維函數 $f(x,y)$ 來描述物體，加上 (θ,t) 參數，我們以線性積分 $P_\theta(t)$ 來描述 (θ,t) 參數對應的 x-ray 射線對物體作投影而得到衰減係數總和。

$$P_\theta(t)=\int_{(\theta,t)} f(x,y)ds \qquad (2.1)$$

使用 delta function

$$\delta(n)=\begin{cases}0, & n\neq 0\\ 1, & n=0\end{cases} \qquad (2.2)$$

公式(2.1)可以改寫如下

$$P_\theta(t)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\delta(x\cos\theta+y\sin\theta-t)dxdy \qquad (2.3)$$

公式(2.3)就是函數 $f(x,y)$ 的雷登轉換(Radon Transform)。集合所有同一個角度θ的 $P_\theta(t)$ 值就成為角度θ的投影資料。

後面將介紹到的濾波反投影演算法是利用傅立葉理論(Fourier Theory)與投影資料來達到前述問題的封閉解(closed form solution)繼而重組出截面圖。連結傅立葉轉換(Fourier Transform)到待測物體的截面圖之間的基礎理論就是傅立葉切片理論(Fourier Slice Theorem)。

2.2 Fourier Slice Theorem

連結傅立葉理論到投影資料與待測物體截面圖之間的基礎理論是由 Bracewell、Ramachandran 和 Lakshminaraynana, 發展，接下來的證明則是由 Kak 和 Slaney 推導的結果。二維傅立葉轉換定義如下：

$$F(u,v)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j2\pi(ux+vy)}dxdy \qquad (2.4)$$

其中 u，v 是 *cycles/unit length*。

接著我們定義角度為θ的投影資料及其傅立葉轉換

$$S_\theta(\omega)=\int_{-\infty}^{\infty}P_\theta(t)e^{-j2\pi\omega t}dt \qquad (2.5)$$

其中ω是 *radians/unit length* 。

為了導出 Fourier Slice Theorem，我們需要定義新的座標系統(t,s)，(t,s)座標是原有的座標系統(x,y)順時針旋轉θ角而來。

$$\begin{bmatrix}t\\s\end{bmatrix}=\begin{bmatrix}\cos\theta & \sin\theta\\-\sin\theta & \cos\theta\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix} \qquad (2.6)$$

參考圖 2-2，圖中的投影資料$P_\theta(t)$是相對於(x,y)座標從 0 度以逆時針旋轉θ角對物體進行投影而得到的資料。若我們考慮新的座標系統(t,s)，則$P_\theta(t)$是相對於(t,s)座標系統以$\theta=0$角度對物體進行投影得到的資料。此時在(t,s)座標系統中，考慮公式

$$P_\theta(t)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}f(x,y)\delta(x\cos\theta+y\sin\theta-t)dxdy$$

及$\theta=0$，沿著常數 t 定義出來的投影可以寫成

$$P_\theta(t)=\int_{-\infty}^{\infty}f(t,s)ds \qquad (2.7)$$

將(2.7)代入(2.5)中我們可以得到

$$S_\theta(\omega)=\int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty}f(t,s)ds\right]e^{-j2\pi\omega t}dt \qquad (2.8)$$

將整理公式(2.8)，以公式(2.6)將(t,s)座標系統轉回(x,y)座標系統，則我們可以得到下列公式

$$S_\theta(\omega)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}f(x,y)e^{-j2\pi\omega(x\cos\theta+y\sin\theta)}dxdy \qquad (2.9)$$

公式(2.9)的右手邊就是 2 維傅立葉轉換的定義，因此

$$S_\theta(\omega)=F(u,v) \qquad (2.10)$$

從公式(2.10)可知道，在θ角度對待測物體截面圖作投影得到的投影資料，經過一維傅立葉轉換後相當於待測物體截面圖的二維傅立葉頻率域(frequency domain)中的一條通

過低頻中心點的射線。圖 2-3 可以清楚的表示公式(2.10)的意義

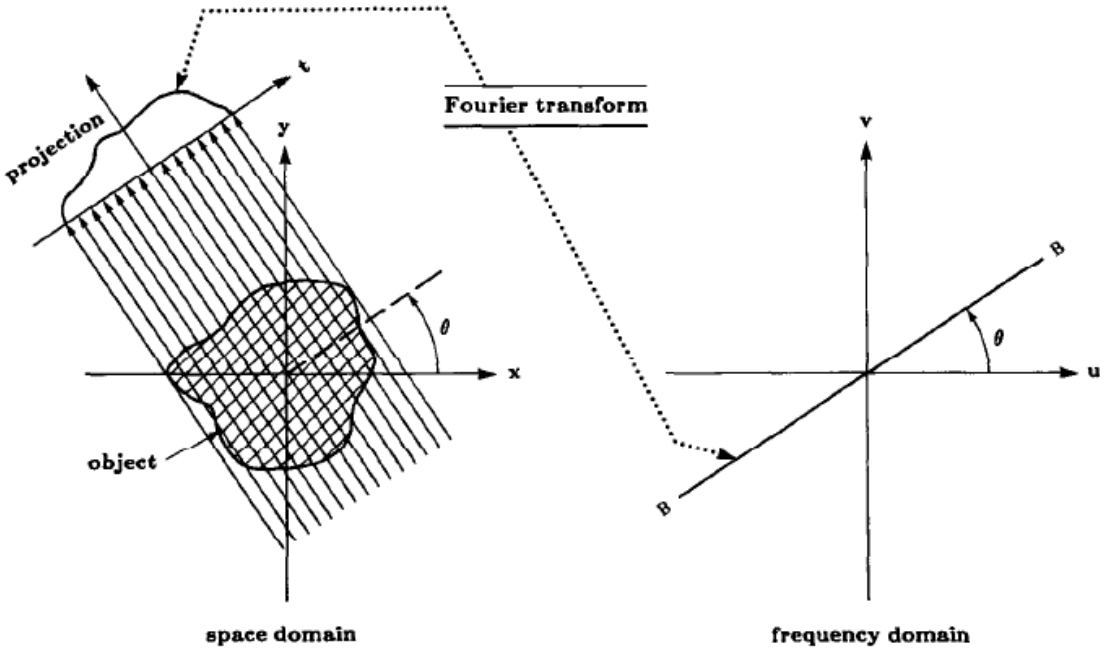


圖 2-3

有了上述結果，我們可以在θ_1，θ_2，...，θ_k角度對物體作投影，得到投影資料後經過一維傅立葉轉換對應到截面圖的二維傅立葉頻率域中。如圖 2-4 所示：

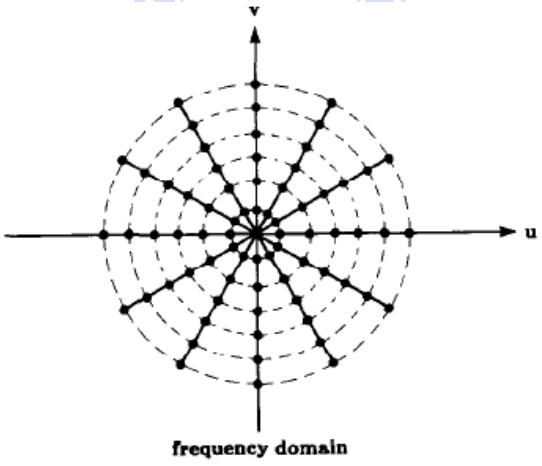


圖 2-4

假設我們可以作無限個不同角度的投影，那我們得到的資料經過一維傅立葉轉換後，可以填滿截面圖的頻率域，接著我們只要再作反傅立葉轉換(inverse Fourier Transform)就可以得到我們所希望的截面圖。

可是實際上，我們只能得到有限個不同投影角度的投影資料，且由圖 可知，從投影資料經過一維傅立葉轉換對應到頻率域中的資料點是中心點(低頻)密集，外圍(高頻)離中心點越遠越稀疏，因此頻率域沒有對應到的資料點就需要從已知資料點作內插法來當成預設資料。可是這其中會有相當大的誤差，尤其是越高頻的地方誤差越大，造成重建回來的截面圖會有影像剝蝕(image degradation)的情形。因為上面提到的是電腦斷層掃描重建影像概念上的模式，實作上我們得用不同的方式來完成。

2.3 濾波反投影演算法

這邊我們要使用來完成影像重建的演算法是濾波反投影(Filtered Back Projection，FBP[(1)])演算法。這演算法也是由Fourier Slice Theorem衍生而來，是藉由公式原理的改寫來達成不同的運算實作。這邊我們先了解濾波反投影演算法的想法。

由前一小節我們可以知道角度θ的投影資料作一維傅立葉轉換可對應到物體截面圖二維傅立葉頻率域中的一條射線，這相當於截面圖二維傅立葉頻率域乘於一個簡單的濾波器(filter)而得到如下圖(b)所示的情形。

(a) (b) (c)

圖 2-5 這圖顯示從投影資料中得到的頻率域情形。(a)是理想狀況，(b)是經過傅立葉切片理論後得到的狀況，(c)是(b)圖經過濾波器加權後到的狀況，(c)圖中的資料狀況近似於(a)

但是對影像重建而言，我們希望得到的是投影資料的一維傅立葉轉換可以跟上圖(a)

的派外型射線是相當的，這樣我們集合幾個這樣的射線就可以重建出高品質的影像。對於這想法，採用最接近的方式就是將投影資料的一維傅立葉轉換—$S_\theta(\omega)$，乘以濾波器—圖(a)中的楔形物的寬度。假設在 180°內共執行K次不同角度的投影，則在頻率ω下，楔形物的寬度是$2\pi|\omega|/K$。$S_\theta(\omega)$在經過$2\pi|\omega|/K$的加權後，我們可以得到圖(c)的圖形，這跟圖(a)來比較，具有相近的”質量”。在足夠的投影次數下，集合圖(c)的射線，可以重建出接近圖(a)所重建的影像。而濾波反投影演算法，聞其名可以知道，演算法是分為濾波(filtering)部分：$S_\theta(\omega)$在頻率域乘以濾波器作加權。及反投影(backprojection)部分：將加權後的$S_\theta(\omega)$值重組回影像。底下我們要推導濾波反投影演算法的數學公式。

我們同樣以二維函數$f(x,y)$來描述待測物體，$f(x,y)$的反傅立葉轉換(inverse Fourier Transform)定義如下：

$$f(x,y)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}F(u,v)e^{j2\pi(ux+vy)}dudv \qquad (2.11)$$

將頻率域的標準座標系統轉成極座標系統(polar coordinate system)，如下

$$u=\omega\cos\theta, v=\omega\sin\theta \qquad (2.12)$$

微分細節如下

$$dudv=\omega d\omega d\theta \qquad (2.13)$$

公式(2.11)可以改寫如下

$$f(x,y)=\int_0^{2\pi}\int_0^{\infty}F(\omega,\theta)e^{j2\pi\omega(x\cos\theta+y\sin\theta)}\omega d\omega d\theta \qquad (2.14)$$

我們將積分(integral)分為 0°到 180°及 180°到 360°兩部分，則

$$\begin{aligned}f(x,y)=&\int_0^{\pi}\int_0^{\infty}F(\omega,\theta)e^{j2\pi\omega(x\cos\theta+y\sin\theta)}\omega d\omega d\theta\\&+\int_0^{\pi}\int_0^{\infty}F(\omega,\theta+\pi)e^{j2\pi\omega[x\cos(\theta+\pi)+y\sin(\theta+\pi)]}\omega d\omega d\theta\end{aligned} \qquad (2.15)$$

利用傅立葉原理

$$F(\omega,\theta+\pi)=F(-\omega,\theta) \tag{2.16}$$

可以簡化公式(2.15)如下

$$f(x,y)=\int_0^{\pi}\left[\int_{-\infty}^{\infty}F(\omega,\theta)|\omega|e^{j2\pi\omega t}d\omega\right]d\theta \tag{2.17}$$

其中

$$t=x\cos\theta+y\sin\theta \tag{2.18}$$

將公式(2.17)中括號的二維傅立葉轉換$F(\omega,\theta)$以投影資料的一維傅立葉轉換$S_\theta(\omega)$來代替，我們得到

$$f(x,y)=\int_0^{\pi}\left[\int_{-\infty}^{\infty}S_\theta(\omega)|\omega|e^{j2\pi\omega t}d\omega\right]d\theta \tag{2.19}$$

整理公式(2.19)中的積分式子

$$f(x,y)=\int_0^{\pi}Q_\theta(x\cos\theta+y\sin\theta)d\theta \tag{2.20}$$

其中

$$Q_\theta(t)=\int_{-\infty}^{\infty}S_\theta(\omega)|\omega|e^{j2\pi\omega t}d\omega \tag{2.21}$$

其中公式(2.21)描述一個濾波的動作。在頻率域中使用$|\omega|$當濾波器對$S_\theta(\omega)$進行加權，所以公式(2.21)稱為濾波投影(filtered projection)。而公式(2.20)則對每個$Q_\theta(t)$進行背投影(back projection)的動作。對於重建影像上的每個點(x,y)，在角度θ下可以找到對應的值$t=x\cos\theta+y\sin\theta$。如圖，由$\theta_i$角可以找到對應的$Q_{\theta_i}(t)$，在$Q_{\theta_i}(t)$距離投影資料中心點$t_j$置，可以找到LM直線上所有$(x,y)$點累加的衰減係數總和$Q_{\theta_i}(t_j)$值，$Q_{\theta_i}(t_j)$值可以平均分布到LM直線上的所有$(x,y)$點。假設在180°內對物體的投影有K次，則對重建影像的點(x,y)而言，每個點可以累加K次對應的$Q_{\theta_i}(t_j)$值，計算出每個點的線性衰減係數，而重建出影像。公式(2.20)及(2.21)就是從投影資料到整個影像重建完成的計算過程。底下我們將介紹在個人電腦上如何以軟體實作此演算法。

備註 1：後面的章節中，濾波反投影演算法將以 FBP 演算法來簡稱。

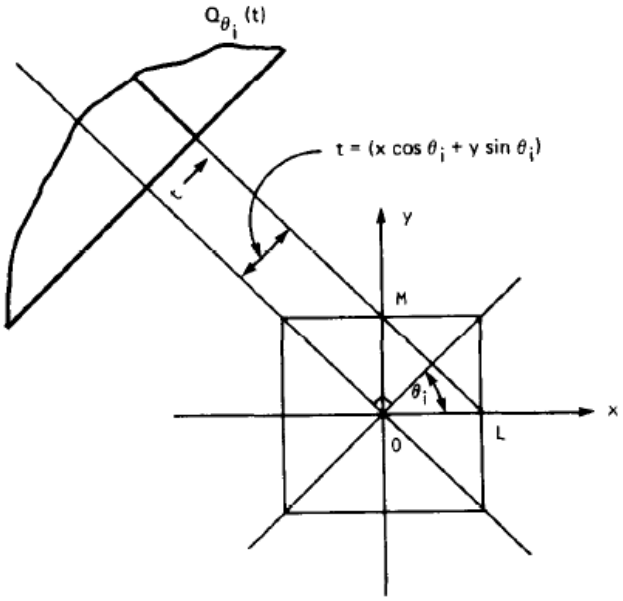


圖 2-6 背投影過程示意圖

2.4 濾波反投影演算法實作

FBP 演算法由前一小節可以知道，共分為濾波投影及反投影兩部分。根據

Filtered Projection $$Q_\theta(t) = \int_{-\infty}^{\infty} S_\theta(\omega)|\omega| e^{j2\pi\omega t} d\omega \qquad (2.20)$$

Back Projection $$f(x,y) = \int_0^{\pi} Q_\theta(x\cos\theta + y\sin\theta) d\theta \qquad (2.21)$$

兩程式，假設 0°到 180°之間對待測物體共作K次不同角度的投影，所以演算法簡單的流程如下：

I. 濾波投影

取得θ_i的投影資料 $P_{\theta_i}(t)$，i從 0 到 $K-1$，執行下列工作共K次。

i. 將 $P_\theta(t)$作傅立葉轉換到頻率域，成為 $S_\theta(\omega)$。

ii. 對$S_\theta(\omega)$乘以濾波器$|\omega|$在頻率域作加權。

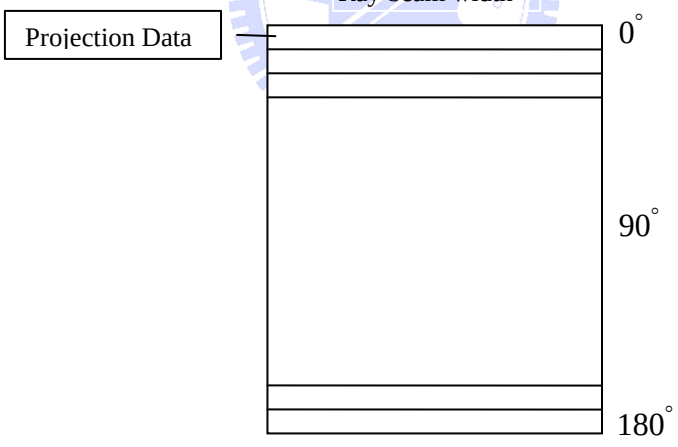
iii. 將加權後的$S_\theta(\omega)$值作反傅立葉轉換回空間域(space domain)，成為$Q_\theta(t)$。

II. 背投影

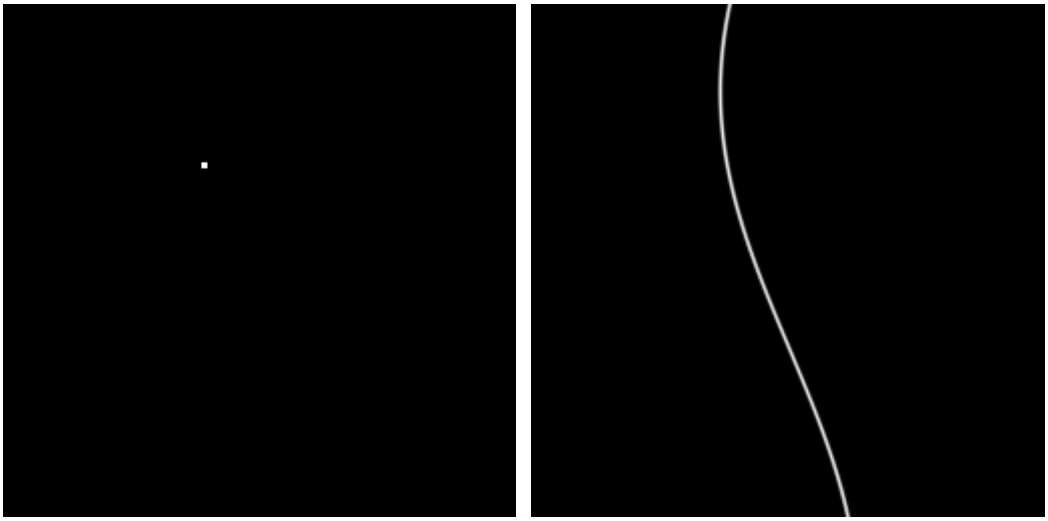
i. 從點(x,y)及角度θ_i算出$t_j = x\cos\theta_i + y\sin\theta_i$，i從 0 到$K-1$，共K次。

ii. 累加每個$Q_{\theta_i}(t)$中，位置t_j上得到的$Q_{\theta_i}(t_j)$值，共累加K次，得到(x,y)點原本的線性衰減係數。

在實作上，電腦斷層掃描取得的投影資料會排列成一張影像，我們稱為sinogram。sinogram的排列，是按照 0°到 180°依序將投影資料以一列一列的方式擺列成一張影像，其中若平行投影光束有 256 束，則每列就有 256 筆偵測器得到的偵測資料。sinogram示意圖如圖 (a)所示：



(a)



(b) (c)

圖 2-7 (a)是 sinogram 影像的示意圖，(b)是 256x256 pixels 灰階的範例圖，(c)是(b)的 sinogram

圖 2-7(b)是 256x256 的灰階影像，其中以座標(100,80)為中心點有個 3x3 pixels的小正方形。而圖 (c)就是對圖 (b)進行 256 次投影，每次投影偵測到 256 筆偵測資料而得到的sinogram影像，因此sinogram影像也是 256x256 pixels大小，在此我們重建影像要求得的pixel點衰減係數就是pixel的灰階值。

我們參考前面的提到的 FBP 演算法流程，先在個人電腦上以 C Language 實作此演算法，以驗證 FBP 演算法的正確性。根據演算法流程，FBP 演算法程式設計流程圖如下所示：

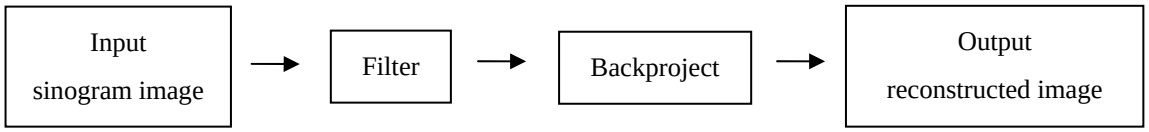


圖 2-8 FBP 演算法程式流程

所以程式上的內容主要也是分成四個部分，以底下的 pseudo code 程式來表示我們

軟體的實作內容：

表 2-1 FBP 演算法的 pseudo code

```
#define angle=128;  /* 旋轉 128 個角度 */
#define rays=128;   /* rays beam 與 image 的 raw pixel 個數一樣 */
const int x_max = rays;
const int y_max = rays;
const int x_cen = x_max/2;  /* x_cen 表示 sinogram 影像 x 軸的中心位置 */
const int y_cen = y_max/2;  /* y_cen 表示 sinogram 影像 y 軸的中心位置 */
double sonogram[angles][ rays];  /* 儲存原始 sinogram 影像的陣列 */
double filtered_rel[angles][rays];  /* 儲存完成 filter 後的 singogra 影像
                                       的陣列 */
double reconstructed[rays][ rays];  /* 儲存重建完成影像的陣列 */
main()
begin
/******從 sinogram 影像中讀取投影資料******************/
    read sinogram image to sinogram[angles][rays];

/******底下是filtered projection的pseudo code **************/
    /* 作 1D FFT 及 IFFT 之前要作 2 冪次方的 ZERO Padding */
    int logn = int(ceil(log10(rays)/log10(2))+1);
    int fftsize = int(pow(2,logn));
    /* 定義FFT及IFFT中存實部，虛部的array */
    double *rel = new double [fftsize+1];
    double *img = new double [fftsize+1];
    /* 定義 window值 */
    double *window = new double [fftsize+1];
    for(i=1; i<fftsize/2; i++)
        window[i+1] = window[fftsize-i+1] = i;
    window[0] = window[1] = 1;
    window[fftsize/2+1] = fftsize/2;

    for( a=0 ; a<angle ; a++ ) loop
        memset(rel,0,sizeof(double)*(fftsize+1));
        memset(img,0,sizeof(double)*(fftsize+1));
        memcpy(rel, sinogram[a],sizeof(double)*rays);

        FFT(rel, img);

        /* 做完 FFT 後與 windows 相乘 */
        for(i=1;i<fftsize+1;i++) loop
            rel[i]*=(window[i]/(double)fftsize*2);
            img[i]*=(window[i]/(double)fftsize*2);
        end loop;

        IFFT(rel, img);

        memcpy(filtered_rel[a],rel,sizeof(double)*rays);
    end loop;

/******底下是back projection的pseudo code **************/
    /* 建立三個tables    查表用 */
    for(x=0; x<angles; x++)
    {
        block_x_cen = block_size;
```

```
        block_y_cen = block_size;
        theta = x*180/angles;

        if(theta<=90)
       {
            sinValue = sinTable[theta];
          cosValue = sinTable[90-theta];
       }
       else
       {
            sinValue = sinTable[180-theta];
           cosValue = 0-sinTable[theta-90];
        }

        shift_x = sinValue;
        shift_y = cosValue;

        x_sino_oricoor = shift_mp - mp*shift_y ;
        y_sino_oricoor = shift_mp + mp*shift_x ;
        x_sino_endcoor = shift_mp + mp*shift_y ;
        y_sino_endcoor = shift_mp - mp*shift_x ;

        u_vector_x = block_x_cen - x_sino_oricoor;
        u_vector_y = block_y_cen - y_sino_oricoor;
        v_vector_x = x_sino_endcoor - x_sino_oricoor;
        v_vector_y = y_sino_endcoor - y_sino_oricoor;

        first_block_sinopoint[x] = (int)(((u_vector_x *
         (v_vector_x/1024)) + (u_vector_y *
         (v_vector_y/1024)))/(128));
        mod_block_x_cen = block_x_cen ;
        mod_block_y_cen = block_y_cen ;

        x_coor_array[x] = mod_block_x_cen + shift_out * shift_x +
((-1)*block_sino_shift*shift_y);
        y_coor_array[x] = mod_block_y_cen + shift_out * shift_y -
((-1)*block_sino_shift*shift_x);
    }
   /*區塊式影像重建*/
   for(block_up=0; block_up<x_max; block_up+=block_size)
    {
        count_y++;
        count_x=-1;
        for(block_left=0; block_left<y_max; block_left+=block_size)
        {
            block_right = (block_left +block_size);
            block_down  = (block_up   +block_size);

            count_x++;
            for(x=0; x<angles; x++)
            {
                  theta = x*180/angles;
               if(theta<=90)
             {
                     sinValue =   sinTable[theta];
                   cosValue  =  sinTable[90-theta];
             }
             else
```

```
            {
                    sinValue = sinTable[180-theta];
                  cosValue = 0-sinTable[theta-90];
               }
               shift_x = sinValue;;
               shift_y = cosValue;

               block_sino_cen = first_block_sinopoint[x] +
               ((shift_y*count_x - shift_x*count_y)*block_size);
          block_sino_start = (block_sino_cen - block_sino_shift);
          block_sino_end   = (block_sino_cen + block_sino_shift);

               x_coor = x_coor_array[x];
               y_coor = y_coor_array[x];

    for(delta_r=block_sino_start to block_sino_end)
   {
         x_coor = x_coor + shift_y;
          y_coor = y_coor - shift_x;
           for(delta_k=0 to shift_num)
           {
                out_x_coor = ((x_coor - (delta_k * shift_x))/1024);
             out_y_coor = ((y_coor - (delta_k * shift_y))/1024);

               if(out_x_coor & out_y_coor in the boundary)
               {

               result[block_size*count_y+out_y_coor][block_size*cou
               nt_x+out_x_coor] += filtered_rel[x][delta_r];
            }
         }
     }
  }
}

/******將重建好的截面圖pixel灰階值寫回重建影像中******************/
    write reconstructed[rays][ rays] to reconstructed imgae;

end;
```

上面的pseudo code內容是針對 128x128 大小的sinogram影像(0°到 180°共 128 個投影角度，每次投影偵測有 128 筆偵測資料)可以重建出截面圖為 128x128 pixels大小的的灰階影像。pseudo code中，在濾波投影(filtered projection)過程時，傅立葉轉換我們以快速傅立葉轉換(Fast Fourier Transform，FFT)來完成，且執行快速傅立葉轉換前，取樣資料需先執行Zero Padding擴大成 2 的冪次方。而在頻率域中需使用的濾波器$|\omega|$函數，則如圖 所示，將投影資料的一維傅立葉值—$S_\theta(\omega)$，乘與一個權重，低頻中心點權重值低，越往高頻則權重越高。

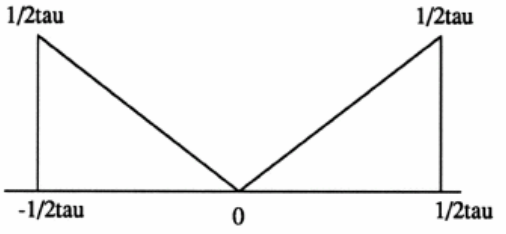


圖 2-9 頻率域中的濾波函數

第三章　區塊化重建影像

本章我們設計一個區塊式重建影像的概念，作為整章闡述的主體，經由實作證實去我們設計的演算法，以下將逐步說明區塊化重建影像概念：

3.1　反投影影像重建概念

由於 ML310 所提供的硬體資源不足以保存整個重建結果，所以必須將重建過程分塊處理，我們採用 4x4 的遮罩作為重建區塊，依序重建出結果，當每個區塊都重建完成後，即為最後的重建影像。如圖 3-1 所示，原圖大小為 128x128，每個區塊大小為 4x4：

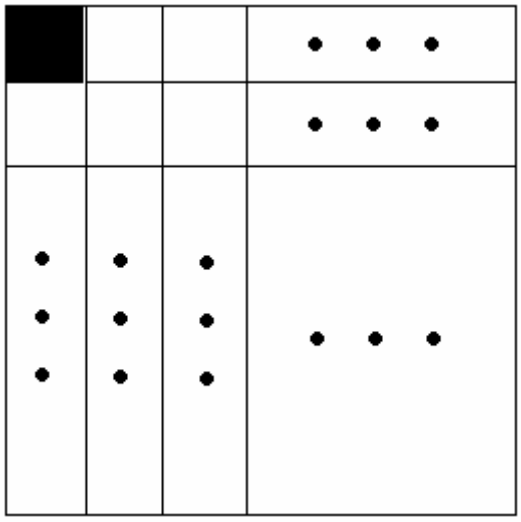


圖 3-1

利用 X-ray 穿透待測物之後得到的衰減值，對應的 x-ray 射線透射物體時直線經過的點的衰減係數總和，如圖 3-2 所示，

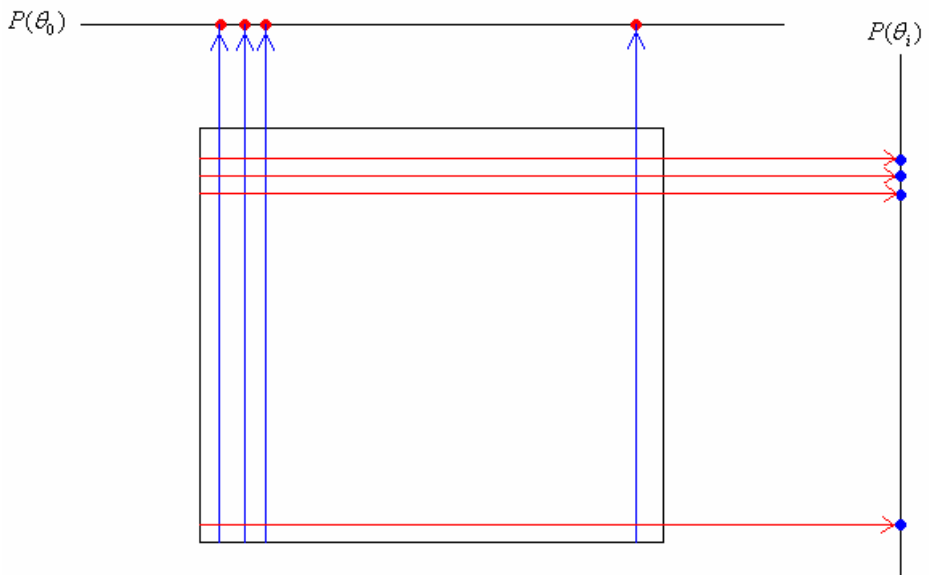


圖 3-2 投影示意圖

相當於是 x-ray 射線經過物體路徑的線性積分。如圖 3-3 所示將 $P(\theta_0) \sim P(\theta_{127})$ 由上而下排列即為 projection data（sinogram）：

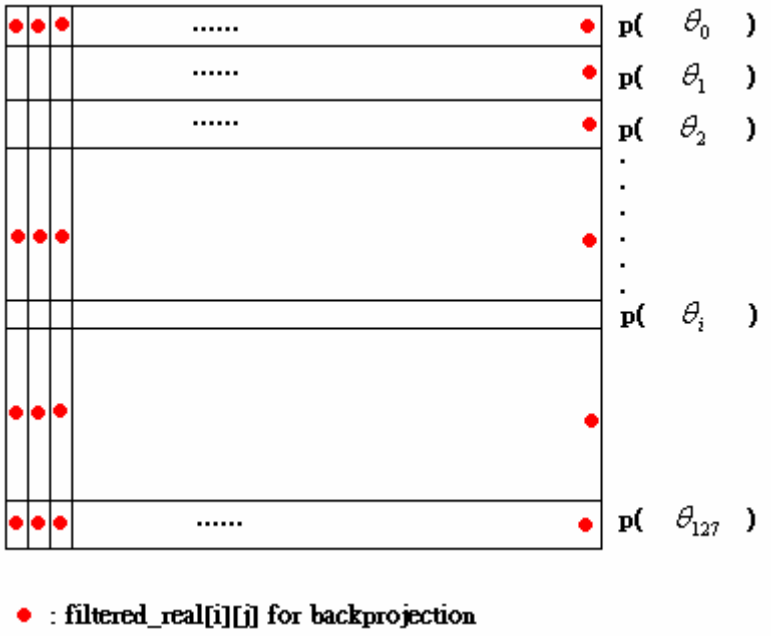


圖 3-3

圖 3-4 為 sinogram，圖 3-5 所示為 filtered sinogram：

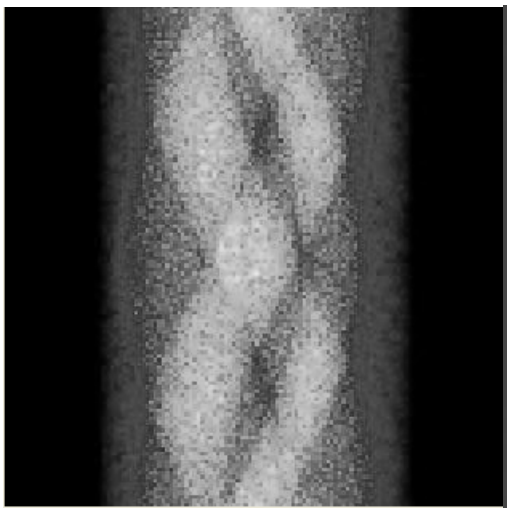


圖 3-4 sinogram

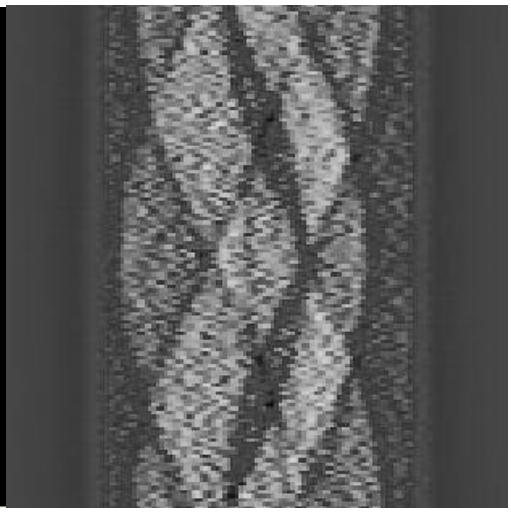


圖 3-5 filtered sinogram

3.2 反投影過程介紹

以下介紹完整 FBP 流程，圖 3-6 所示為整個濾波反投影的流程，對 sinogram 做傅立葉轉換、乘以濾波器、反傅立葉轉換、反投影

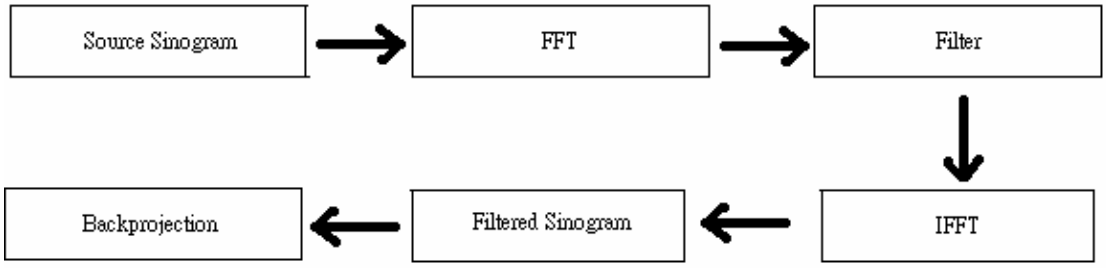


圖 3-6 FBP 流程

經過反傅立葉轉換後，即可得到 filtered-sinogram，之後開始做反投影，我們只取實數部份作反投影，因此將 $P(\theta_i)$ 上的點稱作 $filtered\_real(i, j)$。i 表示為第 i 個角度，即為 $P(\theta_i)$， j 表示在 $P(\theta_i)$ 上第 j 個點，各個 $filtered\_real(i, j)$ 反投影回去時，將 $filtered\_real(i, j)$ 累加在重建影像的像素中，就如同光線投射過去，所經過的區域賦予相同的 $filtered\_real(i, j)$ 值，如圖 3-7 所示：

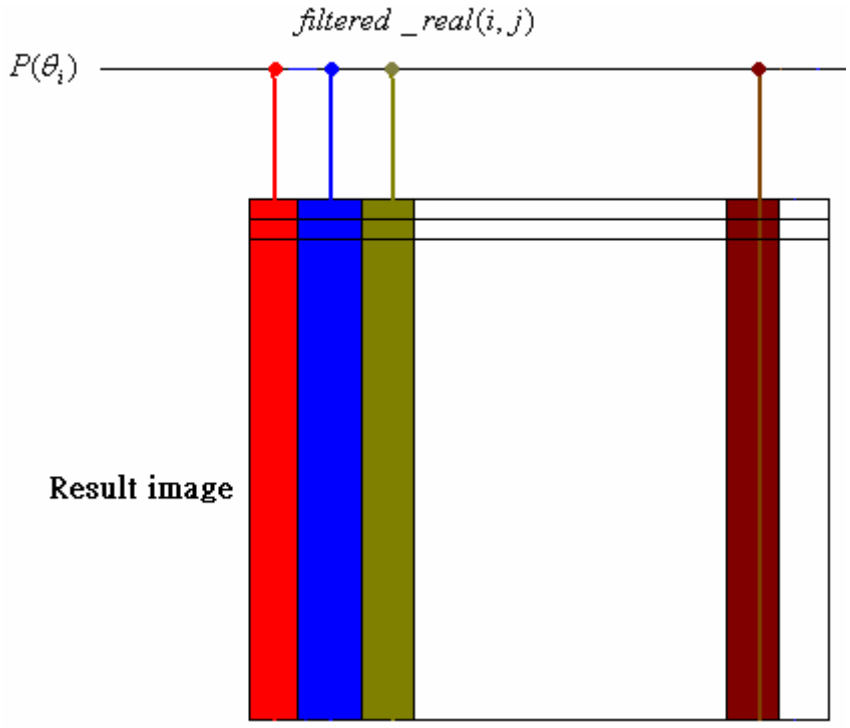


圖 3-7 Backprojection 示意圖

3.3 演算過程推演

當光線直接反投影回去，直接將 $filtered\_real(i,j)$ 累加在重建影像的像素中，此時我們取整數點當作反投影時所經過的座標，當 $filtered\_real(i,j)$ 向前傳播 Δk 時，X 方向的變化量為 $\Delta k \times \sin\theta$ ，Y 方向的變化量為 $\Delta k \times \cos\theta$ ，如此可以方便計算出 $filtered\_real(i,j)$ 下一個將要經過的座標，Δk 每次遞增值為一，以影像為例，左上角為原點，因此，如圖 3-8 所示：

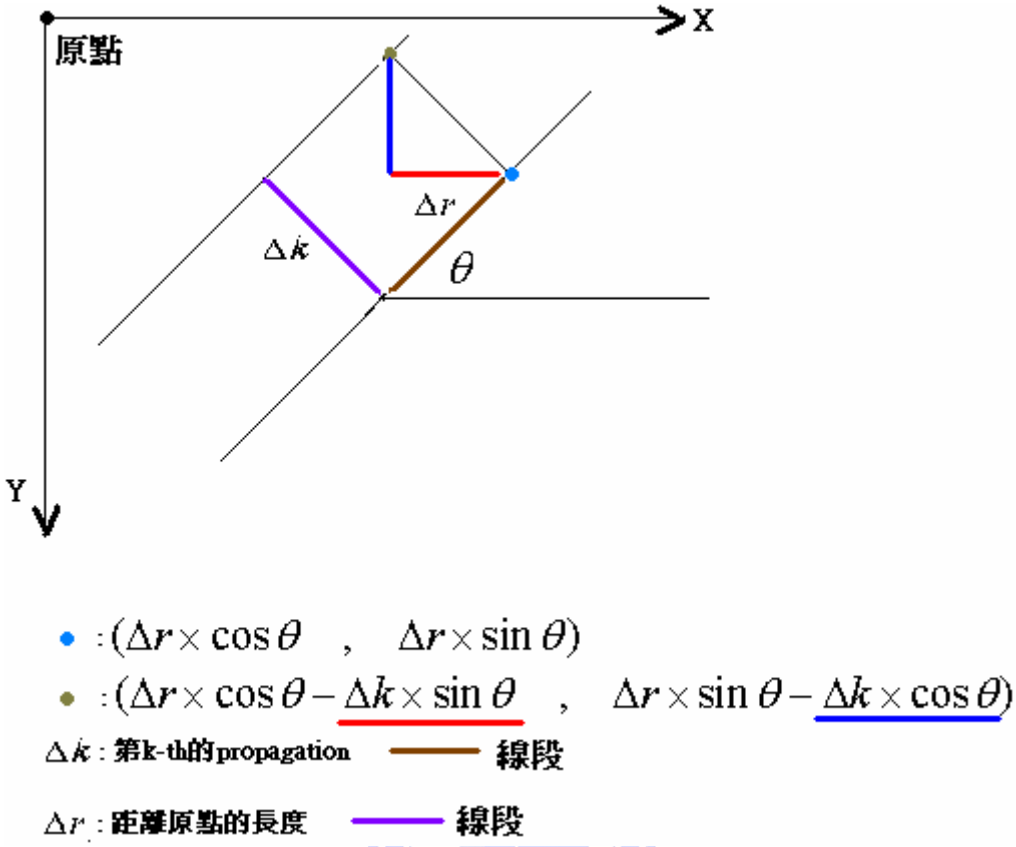


圖 3-8

以直角座標系統為例，O 為原點，P 為距離原點 Δr 的點，O_1 為原點 O 下一個將經過的點座標，P_1 為 P 點下一個將經過的點座標，所以 $\Delta k = 1$，此四點座標分別為下，參照圖 3-9:

$$O(0,0)$$
$$O_1(-\sin\theta, \cos\theta)$$
$$P(\cos\theta, \sin\theta)$$
$$P_1(\cos\theta - \sin\theta, \sin\theta + \cos\theta)$$

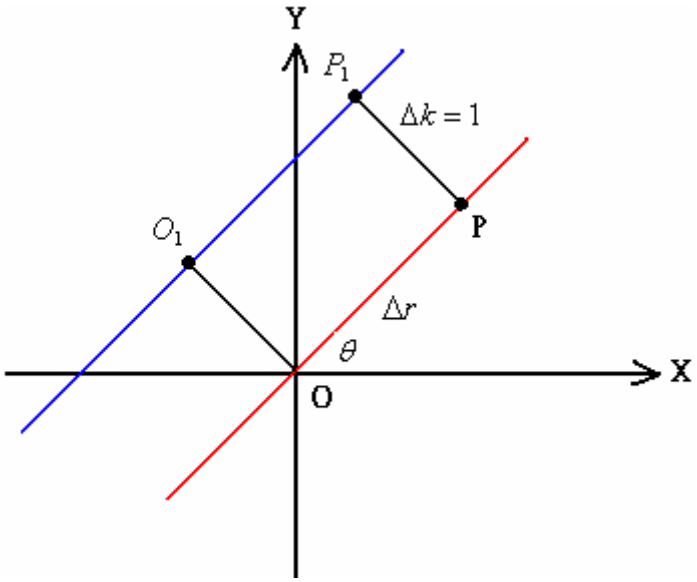


圖 3-9

我們採用區塊式做影像重建，所以每一個區塊只需截出一小段的 sub-sinogram 來做反投影。我們先求區塊中心(即 a 點)、 $filtered\_real(i,0)$(即 b 點)、 $filtered\_real(i,255)$(即 c 點)三點座標，利用 a、b 兩點計算出 $\vec{u}$ 向量，b、c 兩點計算出 $\vec{v}$ 向量，接著利用 Formula 1 計算 $\vec{u}$ 在 $P(\theta_i)$ 上的投影長度 (α)，算出 α 之後截出前後三個單位長，共包含七個 $filtered\_real(i, j)$，即為所需的 sub-sinogram，如圖 3-10 所示：

參照下列公式：

$$\vec{u} = a - b \qquad \vec{v} = c - b$$

$$\alpha = \left\|\vec{v}\right\| \times \cos\theta = \left\|\vec{v}\right\| \times \frac{\vec{u} \cdot \vec{v}}{\left\|\vec{u}\right\|\left\|\vec{v}\right\|} = \frac{\vec{u} \cdot \vec{v}}{\left\|\vec{v}\right\|}$$

$$\beta = \left\|\vec{v}\right\| \times \cos\theta = \left\|\vec{v}\right\| \times \frac{\vec{u} \cdot \vec{v}}{\left\|\vec{u}\right\|\left\|\vec{v}\right\|} = \frac{\vec{u} \cdot \vec{v}}{\left\|\vec{v}\right\|}$$

Formula 1

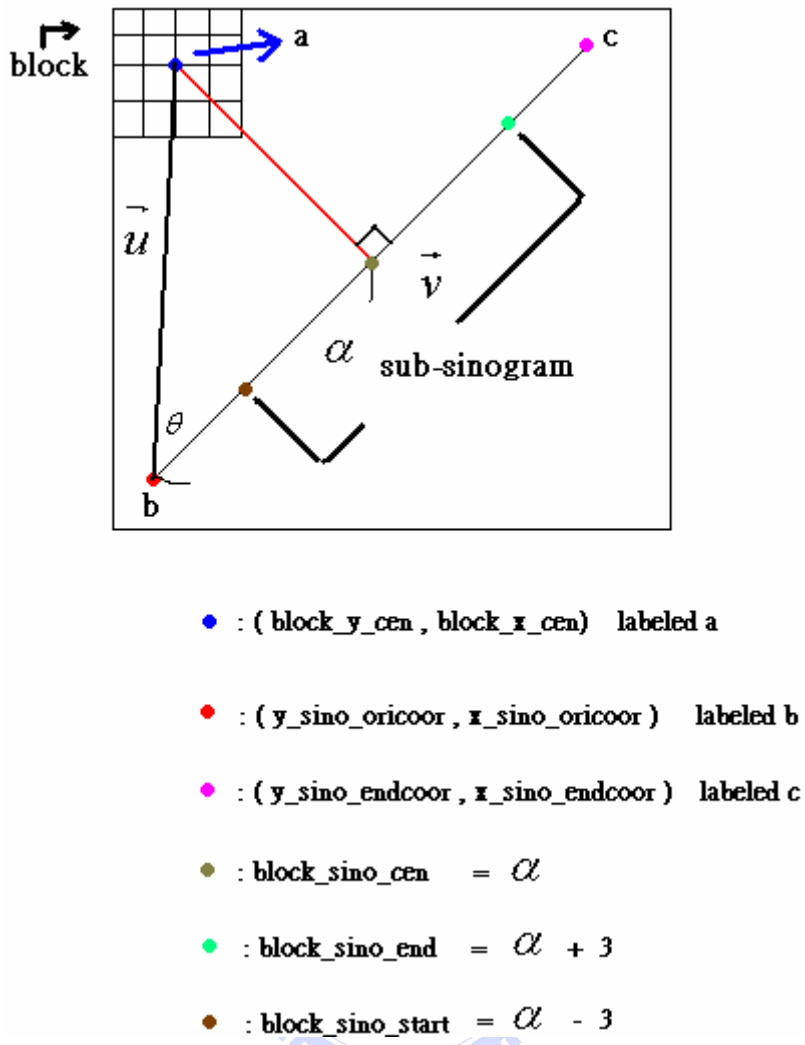


圖 3-10

3.4 拆解區塊式 FBP

我們預先將反投影所需的資訊編成訊息列（message list），每筆訊息長度為 64 位元，訊息內容包括 *filtered _real*(i, j)、Y 座標、X 座標，各佔 32 位元、16 位元、16 位元。如圖 3-11 所示：

filtered _real(i, j)	y_coordinate	x_coordinate
32 bits	16 bits	16 bits

圖 3-11

將七個訊息分別編成上述格式之後，再以平行的方式掃過整個區塊，被掃過的區域立即累加 $filtered\_real(i,j)$ 值，如圖 3-12，由 A 端掃到 B 端，A 端與 B 端的距離為 8 個單位長，每條訊息所走過的路徑長均為 8，如黑色訊息所掃過的區域共有五塊。

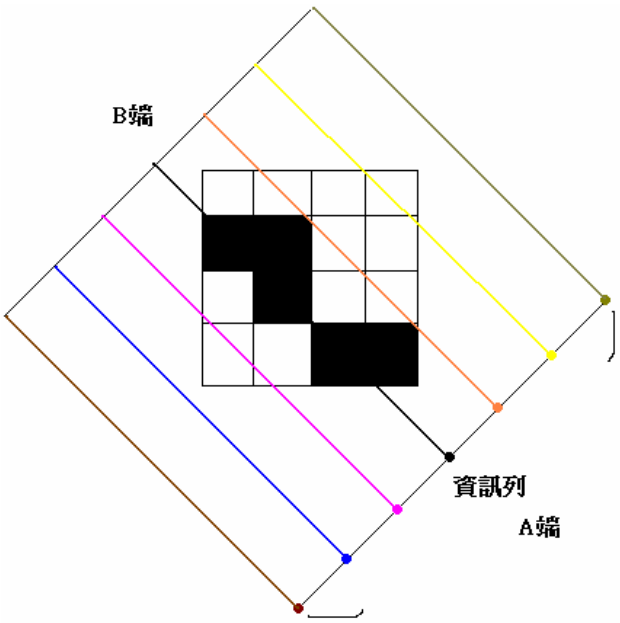


圖 3-12

建立三個表格存放重要的資訊，利用查表法可以更快速將重覆且複雜的計算簡化成表格，用簡單的位移即可計算出其他區塊所需的資訊，我們用的圖大小為 128x128，每個區塊大小為 4x4，因此每一列可以切成 32 個區塊，總共有 1024 個區塊，$0 \le m \le 31$，$0 \le n \le 31$，我們建立的表格資訊皆由 block(0,0)計算得來的，有了 block(0,0)資訊，即可以推算出 block(m,n)於重建時所需的資訊，可計算出 block(m,n)中心點對於 $P(\theta_i)$ 的投影點，進而截出反投影時所需的 sub-sinogam，不用每次重建某一個區塊時，就重新計算 sub-sinogram，節省不少計算上的時間，如圖 3-13：

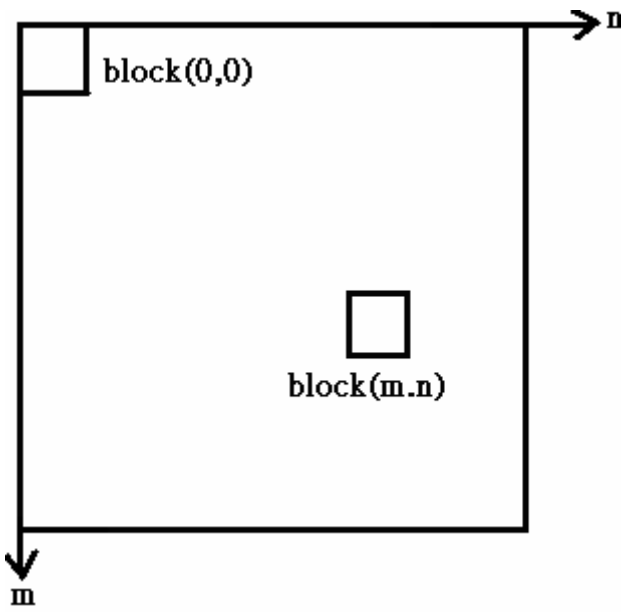


圖 3-13

所要建立的三個表格分別儲存$\alpha_i(0,0)$，$0 \le i \le 127$，截出重建時所需要的 sinogram 之後，第一個點的 X 座標與 Y 座標，即圖 3-14 中標示紅色的點，$\alpha_i(0,0)$ 為 blcok(0, 0) 在 $P(\theta_i)$ 上的投影點：

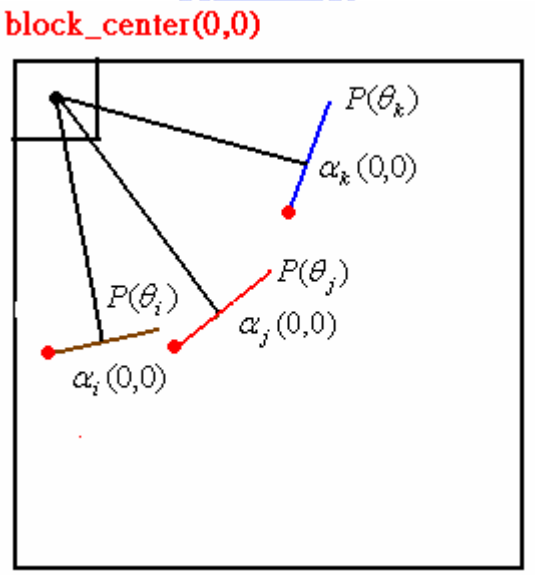


圖 3-14

每個區塊中心點對 $P(\theta_i)$ 的投影點關係，經由推導可得 $\alpha_i(m,n) = 4 \times (n \times \cos\theta_i - m \times \sin\theta_i) + \alpha_i(0,0)$，由此可得到所有區塊中心點於 $P(\theta_i)$ 的投影點，如圖 3-15 所示：

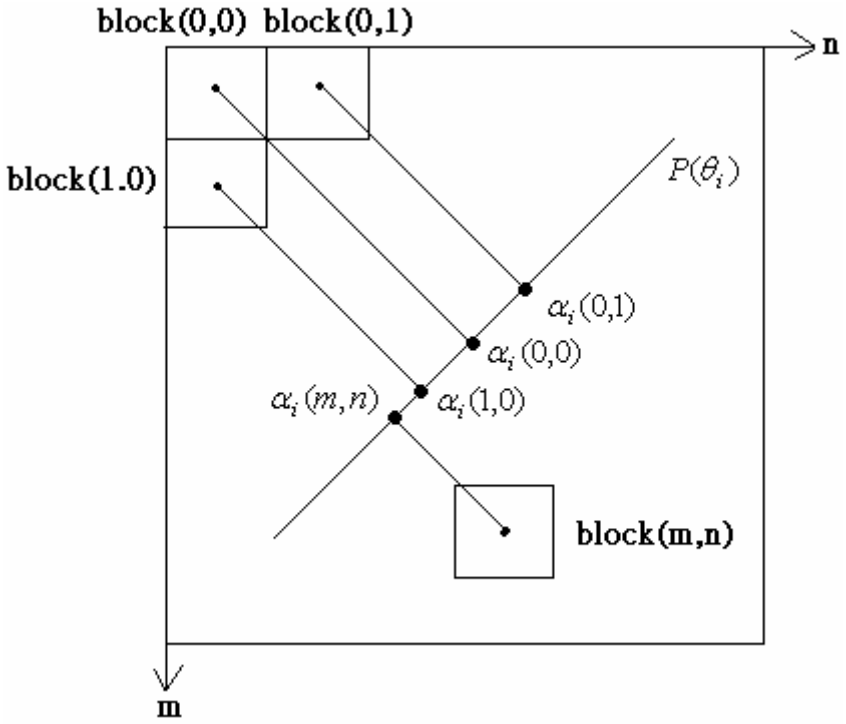


圖 3-15

3.5 如何平行反投影

編好所有訊息之後，即可立即做反投影，可是相鄰的訊息在做反投影的時候，有可能會加到同一個格子點上，造成資料衝突，在同一個時脈同一個格子點上只能有一個被加數，如圖 3-16 所示，紅色訊息和藍色訊息會經過相同的格子點，在同一個時脈同一個格子點中，要累加紅色訊息中的 $filtered\_real(i, j)$，還是要累加藍色訊息中的 $filtered\_real(i, j)$？

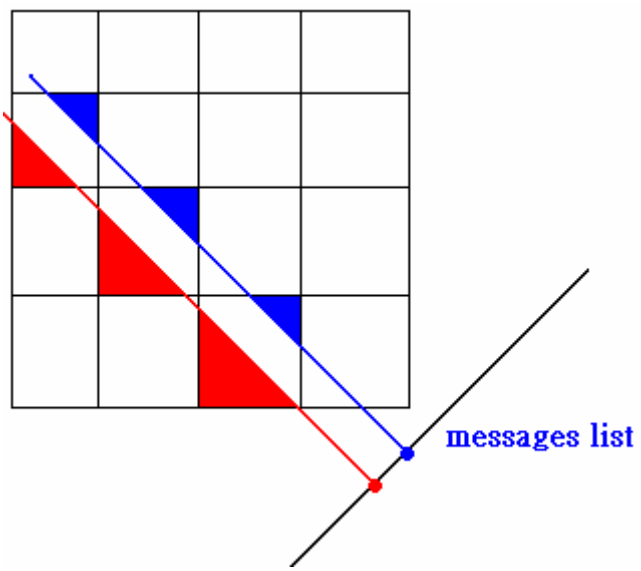


圖 3-16

有鑑於上述情形的發生，我們將訊息列分成偶數訊息(even messages)與奇數訊息(odd messages)，偶數訊息有自己的重建區塊，奇數訊息亦有自己的重建區塊，奇數訊息和偶數訊息重建之後的結果點對點相加起來，即為最後的重建圖形，偶數訊息與奇數訊息於反投影過程中不會有資料衝突的情形，如圖 3-17 所示：

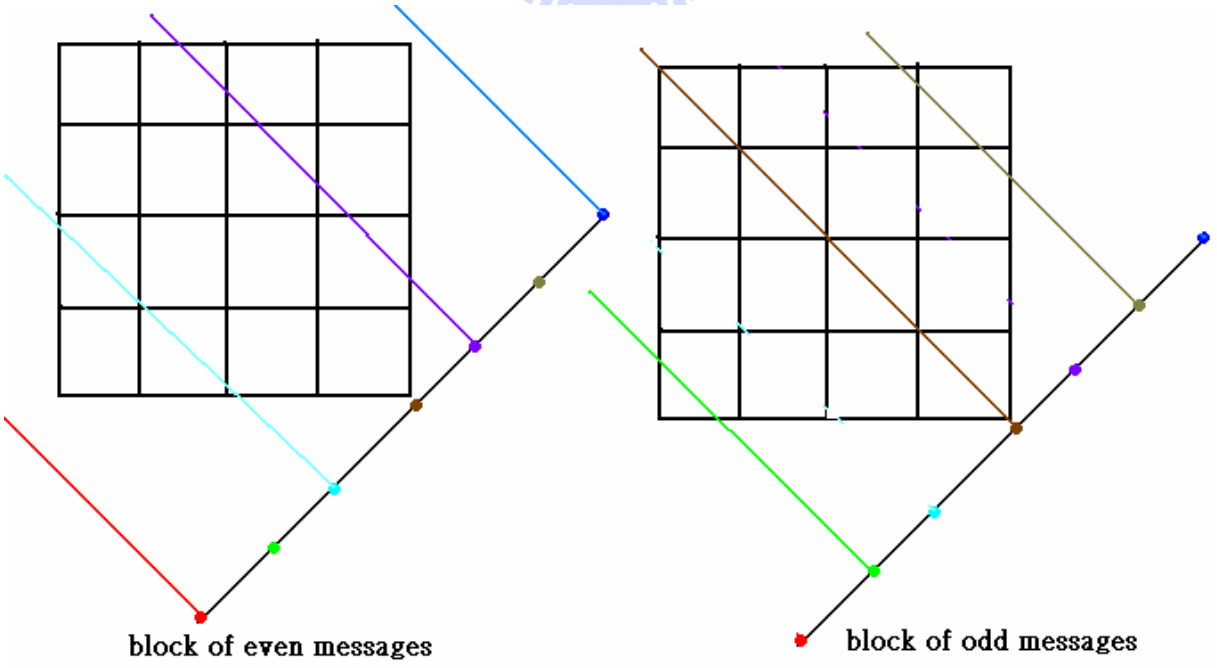


圖 3-17

第四章　結果與討論

4.1 結果與討論

以下為軟體模擬的結果，利用 Xilinx 所開發的一套嵌入式系統開發軟體 EDK(Embedded Development Kit)進行軟體模擬，撰寫 C 語言，搭配 ML310 中的 powerPC 進行運算，時脈為 100MHz，圖 4-14、圖 4-15 分別為 128x128 和 256x256 的原始圖，圖 4-1~圖 4-6 影像大小為 128x128，圖 4-1 採用區塊大小為 4X4 的結果，圖 4-2 採用區塊大小為 8X8 的結果，圖 4-3 採用區塊大小為 16X16 的結果，圖 4-4 採用區塊大小為 32X32 的結果，圖 4-5 採用區塊大小為 64X64 的結果，圖 4-6 採用區塊大小為 128X128 的結果，明顯的圖 4-6 結果較好，因為不需要計算截出 sub-sinogram，可避免誤差累積，其他不同大小的區塊重建結果，由於在截出 sub-sinogram 時，直接取其整數值，亦沒有做內插作為些許的修正，因此結果中出現了假影(artifacts)，此演算法目標用於硬體加速影像重建，我們所使用的硬體資源有限，最後我們採取區塊的大小為 4x4，此演算法可高度平行化的做影像重建。

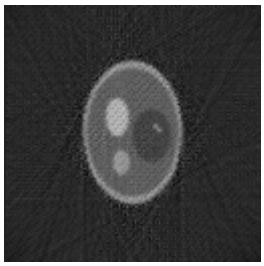


圖 4-1

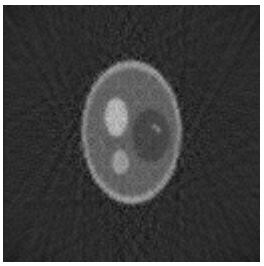


圖 4-2

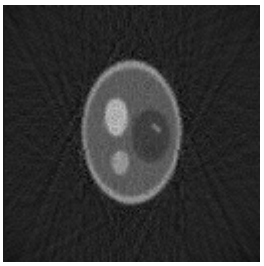


圖 4-3

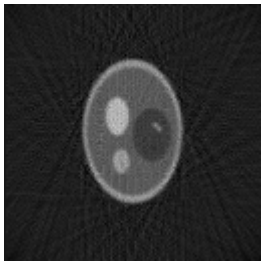


圖 4-4

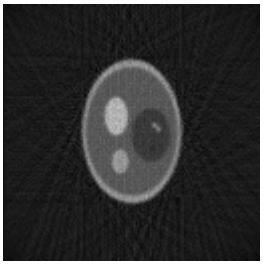


圖 4-5

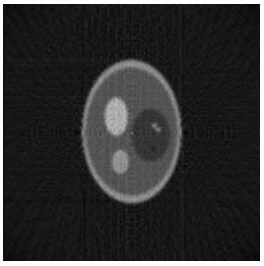


圖 4-6

圖 4-7~圖 4-13 為影像大小 256x256 的軟體模擬結果,圖 4-7 採用區塊大小為 4X4 的結果，圖 4-8 採用區塊大小為 8X8 的結果，圖 4-9 採用區塊大小為 16X16 的結果，圖 4-10 採用區塊大小為 32X32 的結果，圖 4-11 採用區塊大小為 64X64 的結果，圖 4-12 採用區塊大小為 128X128 的結果，圖 4-13 採用區塊大小為 256X256 的結果，相同的道理，圖 4-13 結果較佳，因此若有更多硬體資源的話，可得到較好的重建結果。

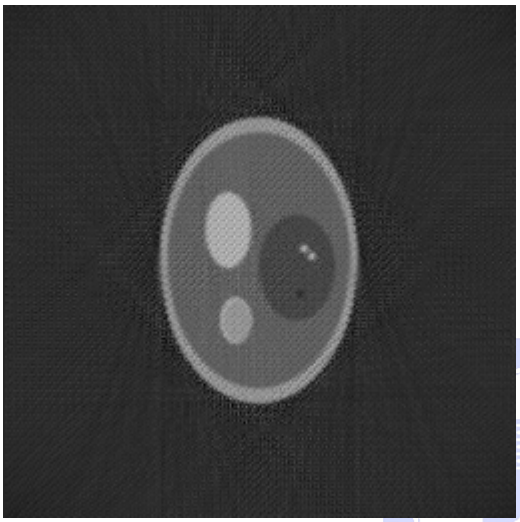


圖 4-7

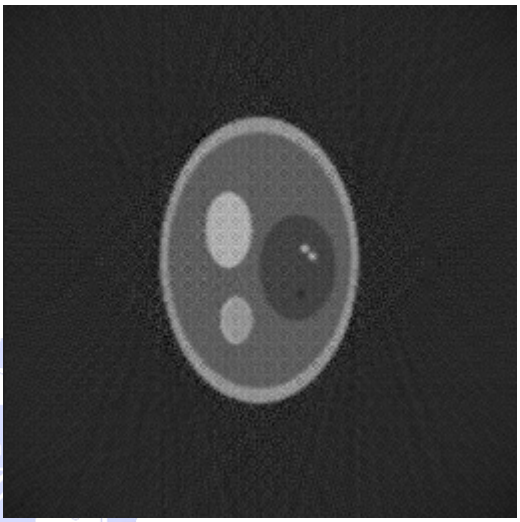


圖 4-8

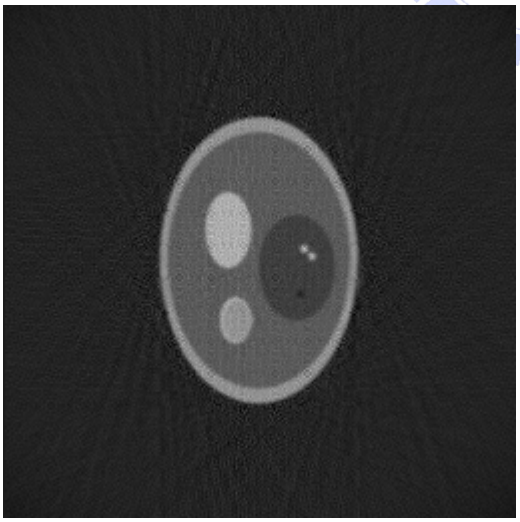


圖 4-9

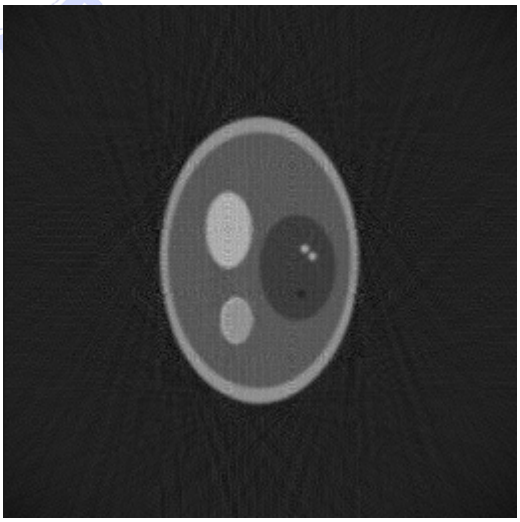


圖 4-10

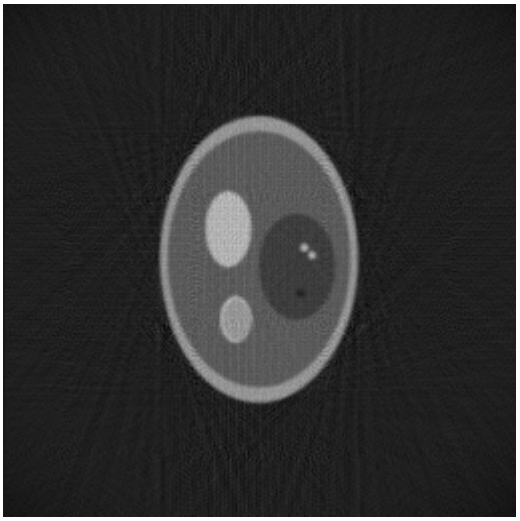


圖 4-11

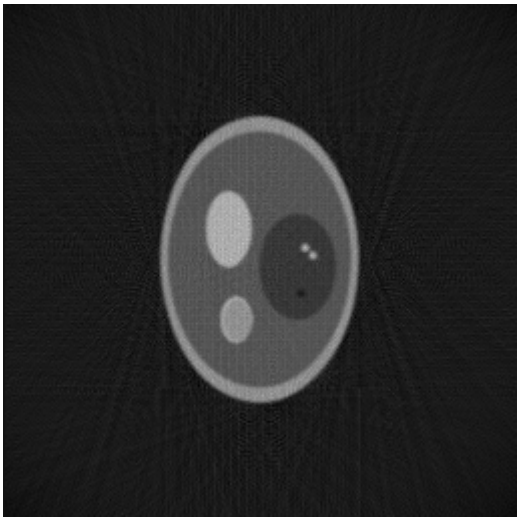


圖 4-12

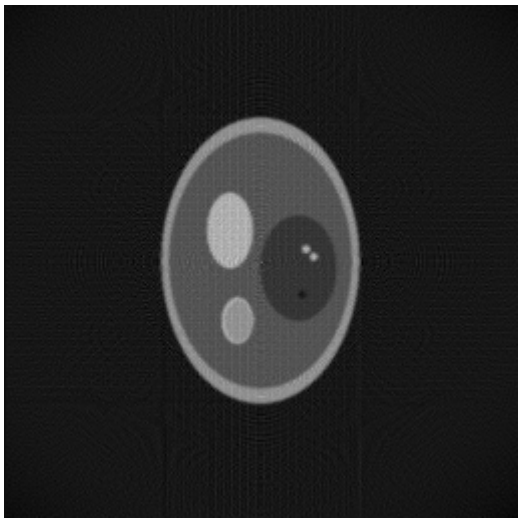


圖 4-13

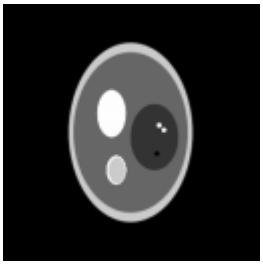


圖 4-14

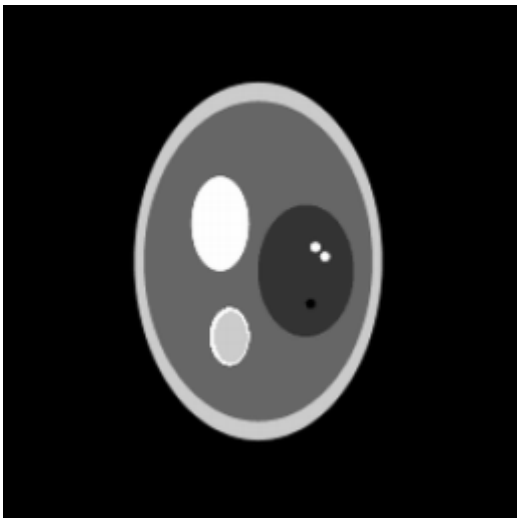


圖 4-15

4.2 未來與展望

本演算法可於 FPGA 硬體加速計算，可高度平行化處理計算，且可以利用 pipeline 排程，當每一列做完傅立葉轉換、乘上濾波器、反傅立葉轉換，即可立刻做反投影，平行化處理加上 pipeline 排程，預期的效果可以更好，執行時間可縮短，更有效率，使硬體效能發揮到極致。

參考文獻

[1] Alexandro M.S. Adario, Eduardo L. Roehe, Sergio Bampi, "Dynamically Reconfigurable Architecture for Image Processor Applications", DAC99, New Orleans, Louisiana, ACM, 1999

[2] Didier LATTARD, Guy MAZARE, "Image reconstruction using an original asynchronous cellular array", ISCAS IEEE, 1989

[3] Anup B. Sharma, Keith R. Allen and Roy P. Pargas, "Some new systolic designs for two-dimensional convolution", ACM, 1988

[4] Bei-Chuan Chen, Yu-Tai Ching, "A new antialiased line drawing algorithm", Computers & Graphics, 2001

[5] Bertil Schmidt, Manfred Schimmler, Heiko Schroder, "Tomographic Image Reconstruction on the Instruction Systolic Array", Computers and Artificial Intelligence, 1995

[6] A.V. Lakshminarayanan, "Reconstruction from divergent ray data," tech. rep., Dept.Computer Science, State University of New York at Buffalo, 1975.