1. **朗伯比尔定律**

简单来说，就是每种物质的核外电子数目都不一样，对光的吸收也不一样。通过检测前后能量的差异可以求出物体对光的吸收能力的大小，也就是衰减系数。把不同的衰减系数对应到不同的像素值，就得到了图像。

1. **投影**

正投影：对投影线经过的像素做线积分，积分得到的值保存为该角度下的权值。

对一组数据 P 做 Radon 变换，即做正投影，会得到两个数据

[R, xp] = radon(P,theta);

xp为积分值对应的径向坐标；

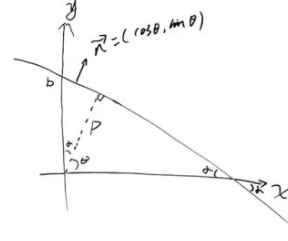
R是theta角下投影线得到的线积分，即权值

反投影：反投影是利用上面投影得到的 R 权值，把R值投回到 x y 坐标中。

x y 满足 x\*cos(theta) + y\*sin(theta) = R 就表明点(x, y)在投影线上 。

注：这个theta 是投影角，不是投影线与x轴夹角，他们之间相差 pi/2，此时把R值全部赋给投影线上的每一个点，最后每个角度的R值叠加到这个点上，再除以总投影线数，就得到了原始数据。

**3．Rodon变换**

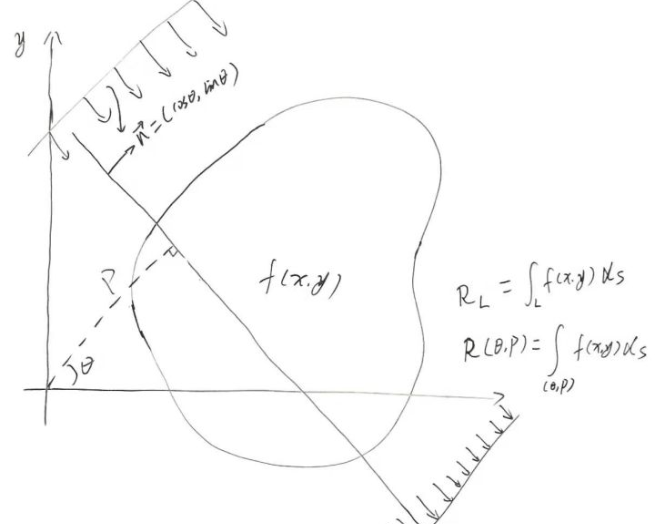


设直线方程为 直线远离原点的法线方向，原点到直线的距离为 ，则如上图公式所示，直线可表示为：



其中该直线斜率 为负数 。

可以看到 即为上图所说的方向，为该方向上的每一点放射源，则根据 可以定义Radon 变换 。



Radon变换相当于把函数通过线积分表示成了另外的一种直线参数的形式，相当于把二维平面 坐标系映射到了直线参数 坐标系**。**



由于线积分把所有点约束在了直线 上，故为了能够展开上式使用了函数，则展开如下：



**也可用表示，雷登变换是投影重建的基石，计算机断层是其在图像处理领域的主要应用，在离散情况下，上式变为：**

****

**4. 傅里叶切片定理**

傅里叶切片定理即投影的一维傅里叶变换和被投影区域图像的二维傅里叶变换间的关系。投影的一维傅里叶变换为：



其中为频率变量，为一定值，由radon变换和上式可得：



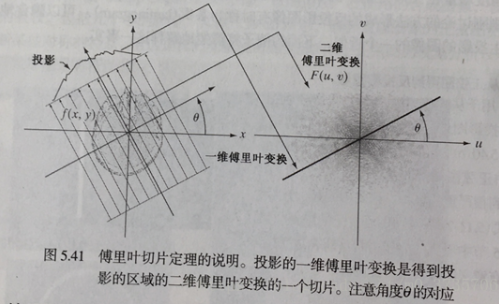
令，带入上式得：



该式为指定，值时的二维傅里叶变换，即：



式中，表示的二维傅里叶变换。



傅里叶切片定理（或投影切片定理）说明了一个投影(一维)的傅里叶是得到这个投影的二维区域的二维傅里叶变换对应角度下的一个切片。正如上图所示，任意一个投影的一维傅里叶变换可以沿着一个角度提取一条直线的的值来得到，而该角度就是投影时所用的角度。

**5. 滤波反投影推导过程**

下面推导滤波反投影公式，将用到傅里叶切片定理。

图像频域的反傅里叶变换为：



令，，通过二重积分换元公式（）将上式表示成极坐标的形式。



然后利用傅里叶切片定理带入：



将的积分范围分解成两个范围：0到180度，180到360度，积分表达式也分解成两个表达式，并利用这一事实：



考虑关于的积分项，注意为一常数，因此可写为：



括号里面的表达式是一个一维傅里叶反变换，附加了一个项，是一个一维滤波函数。是一个斜坡滤波器。这个函数不可积，因为其幅度在两个方向上都扩展到了正无穷，所以其傅里叶变换无定义。理论上可用某种方法来处理该问题，例如使用所谓广义的函数。实践中，该方法是对斜坡加窗，使它的频率范围之外为零。

**6. 滤波器(滤波函数)和内插函数的选取**

由于直接使用反投影算法会存在很不好的因素：

1. 图像模糊
2. 不准确的数据重建图像就会产生各种伪影。
3. 投影的数据是天然离散的，处理不当的话会产生很大的误差。

所以要在计算反投影前对投影做简单的滤波。我们需要将滤波器的频率限定在一个频段范围内，即对滤波器加窗，当使用陡峭的窗函数时会出现明显的振铃现象，所以要用平滑窗代替。

常见的滤波器有R-S滤波函数和S-L滤波函数。R-L滤波函数滤波计算简单，避免了大量的正弦、余弦计算，得到的采样序列分是分段现行的，并没有明显的降低图像质量，所以重建图像轮廓清楚，空间分辨率高。

常见的插值方法有最近邻插值和双线插值，最近邻插值即将离散点中间的缺失值用离它最近的整数处的投影值来替代。

**7. fbp算法步骤**

（1）计算每一个投影的一维傅里叶变换；

（2） 用滤波函数乘以每一个傅里叶变换；

（3）得到每一个滤波后的变换的一维反傅里叶变换；

（4）对步骤3得到的所以一维反变换积分（求和）。

**8. 仿真结果**



**9. 程序：**

P = phantom(128); % 创建一副图像

theta = 1:180; %投影角度

% radon变换获得正投影

[R,xp] = radon(P,theta);

%快速傅里叶变换宽度

width = 2^nextpow2(size(R,1));

%对投影进行快速傅里叶变换

proj\_fft = fft(R, width);

% 滤波 采用RL滤波器

% 构造滤波器

filter = 2\*[0:(width/2-1), width/2:-1:1]'/width;

%滤波 频域相乘 空间域卷积

proj\_filtered = zeros(width,180);

for i = 1:180

proj\_filtered(:,i) = proj\_fft(:,i).\*filter;

end

%滤波后进行一维反傅里叶变换

proj\_ifft = real(ifft(proj\_filtered));

% 反投影

fbp = zeros(128); %像素尺寸

for i = 1:180

rad = theta(i)\*pi/180;

for x = (-128/2+1):128/2

for y = (-128/2+1):128/2

t = round(x\*cos(rad+pi/2)+y\*sin(rad+pi/2));

fbp(x+128/2,y+128/2)=fbp(x+128/2,y+128/2)+proj\_ifft(t+round(size(R,1)/2),i);

end

end

end

fbp = fbp/180;

% 画图

subplot(1, 2, 1), imshow(P), title('Original')

subplot(1, 2, 2), imshow(fbp), title('FBP')