摘 要

本论文较详细地论述和给出了基于塑料闪烁光纤在瞬态低能至高能 X 射线下的CT(Computed Tomography)，DR(Digital Radiography)成像系统的模拟计算、设计原理和实验研究，总结了闪烁光纤阵列用于射线成像方面的最新研究成果。闪烁光纤阵列成像探测器系统综合运用了闪烁光纤探测技术、CCD 光电转换技术、采用滤波反投影法进行图像重建以及计算机模拟技术，有效的实现了在低能和高能下进行射线成像的应用要求。文中介绍了利用 GEANT 系统进行的大量模拟研究工作，给出了闪烁光纤探测特性、闪烁光纤阵列成像探测器的模拟方法和结果，并结合各项实验工作，探讨了一种较为新颖的，具有重要应用价值的成像方法，还就在高能下运用此类探测器的优势以及需要进行改善的地方都进行了深入的分析计算，取得了重要的结果。

采用闪烁光纤成像在射线成像领域是一种较为新颖的方法，而具体本项研究工作的创新点在于：

1) 采用塑料闪烁光纤构造阵列成像探测器用于高能成像，这是一种创新性的思想方法。

2) 研究了闪烁光纤阵列的吸收效率问题，得出了重要的结论。闪烁光纤吸收效率可以通过增加光纤长度来改善，较传统闪烁体非常灵活，但同时又并不是可以一味增长光纤长度以达到提高吸收效率的目的，这其中存在一个光纤长度饱和度问题，即当光纤长度增加到某一数值时，探测器吸收效率几乎不再增加，即达到饱和状态。

3) 研究了采用闪烁光纤阵列所成图像的空间分辨率和对比度问题。较传统闪烁体而言，在相同条件下，光纤阵列的空间分辨率较好，并与探测器厚度基本无关，从而解决了探测器吸收效率与空间分辨率之间存在的矛盾，有利于提高图像信噪比，这就为塑料闪烁光纤阵列作为高能成像探测器奠定了基础。

4) 采用单根较细的闪烁光纤实现了断层扫描成像。这种方法原理装置都非常简单，非常适合科研单位进行小范围成像实验研究时采用。它得到的最后图像空间分辨率主要取决于所采用光纤本身的大小以及被测物体移动的步长。

5) 提出采用在光纤阵列中加入铅等吸收器来减少串扰的方法，并作了系统的模拟分析，得到有价值的结论，这对高能下的成像意义尤为重大。

6) 设计了一个基于塑料闪烁光纤的瞬态 X 射线的二维成像系统，并巧妙的协调实现了在此

条件下发出的瞬态可见光能够被CCD同步采集到。

关键词：塑料闪烁光纤阵列，射线成像，GEANT，图像重建

Abstract

In this thesis we presented the simulating calculation and discussed the designation principle and experiments of the CT and DR imaging systems based on scintillation optical fiber array for the X-ray spectrum from low energy to high energy. It gives also a introduction of the latest research results of radiation imaging using the scintillation optical fiber array. The imaging detection system can realize the application of radiation imaging for low and high X-ray energy based on many advanced technologies, including scintillation optical fiber detection, photon electron transformation with CCD, image reconstruction from projection, and computer simulation, etc.

In this thesis, some research works by using GEANT system are introduced. Methods and results of the detecting property simulation for scintillation optical fiber and scintillation optical fiber array imaging system are described in details. Comparing with the experiment results, we discussed the latest and important imaging method and analyzed deeply the advantages and disadvantages of this kind of detector for incident X-ray with high energy. Furthermore, we got many important scientific results.

The imaging based on scintillation fiber array is a new method in the field of radiation imaging. In our research, we have the following creative points.

1) It's a new idea that we apply the imaging system based on scintillation fiber array to radiation imaging under high energy X-ray.
2) We studied the absorption efficiency of the scintillation optical fiber array and got some important conclusions. The absorption efficiency of the scintillation optical fiber array can be improved by increasing the fiber length. That is an advantage comparing to traditional scintillator. But we can't and needn't increase the fiber length blindly because of saturation of fiber length. That means when fiber length reaches a definite value, the absorption efficiency will not be improved anymore.
3) We studied the contrast and spatial resolution of the imaging produced from radiation imaging system based on fiber. Comparing to traditional scintillator, the resolution of the imaging based on fiber has no relationship with the thickness of detector under same conditions. So the merit solves the tradeoff between the efficiency of detection and spatial resolution of imaging, and is good for improving the signal-to-noise ratio of imaging system.
4) We realized Computerized Tomography by single scintillation fiber. In this imaging method, the device is very simple and is fit for teaching and research organization. The spatial resolution from the system is depend on the fiber size and moving step of measuring system.
5) We put forward the way of controlling the crosstalk among fibers. We found that in order to absorbing the scattering photons we can add lead into fiber array. The simulation adopting this method shows that the imaging quality is improved quit a lot.
6) We designed an imaging system for instantaneous X-ray and solved the problem of visible light captured synchronously by CCD for instantaneous incident photons.

Key words: scintillation optical fiber array, radiation imaging, GEANT, image reconstruction

条件下发出的瞬态可见光能够被CCD同步采集到。

关键词：塑料闪烁光纤阵列，射线成像，GEANT，图像重建

Abstract

In this thesis we presented the simulating calculation and discussed the designation principle and experiments of the CT and DR imaging systems based on scintillation optical fiber array for the X-ray spectrum from low energy to high energy. It gives also a introduction of the latest research results of radiation imaging using the scintillation optical fiber array. The imaging detection system can realize the application of radiation imaging for low and high X-ray energy based on many advanced technologies, including scintillation optical fiber detection, photon electron transformation with CCD, image reconstruction from projection, and computer simulation, etc.

In this thesis, some research works by using GEANT system are introduced. Methods and results of the detecting property simulation for scintillation optical fiber and scintillation optical fiber array imaging system are described in details. Comparing with the experiment results, we discussed the latest and important imaging method and analyzed deeply the advantages and disadvantages of this kind of detector for incident X-ray with high energy. Furthermore, we got many important scientific results.

The imaging based on scintillation fiber array is a new method in the field of radiation imaging. In our research, we have the following creative points.

1) It's a new idea that we apply the imaging system based on scintillation fiber array to radiation imaging under high energy X-ray.
2) We studied the absorption efficiency of the scintillation optical fiber array and got some important conclusions. The absorption efficiency of the scintillation optical fiber array can be improved by increasing the fiber length. That is an advantage comparing to traditional scintillator. But we can't and needn't increase the fiber length blindly because of saturation of fiber length. That means when fiber length reaches a definite value, the absorption efficiency will not be improved anymore.
3) We studied the contrast and spatial resolution of the imaging produced from radiation imaging system based on fiber. Comparing to traditional scintillator, the resolution of the imaging based on fiber has no relationship with the thickness of detector under same conditions. So the merit solves the tradeoff between the efficiency of detection and spatial resolution of imaging, and is good for improving the signal-to-noise ratio of imaging system.
4) We realized Computerized Tomography by single scintillation fiber. In this imaging method, the device is very simple and is fit for teaching and research organization. The spatial resolution from the system is depend on the fiber size and moving step of measuring system.
5) We put forward the way of controlling the crosstalk among fibers. We found that in order to absorbing the scattering photons we can add lead into fiber array. The simulation adopting this method shows that the imaging quality is improved quit a lot.
6) We designed an imaging system for instantaneous X-ray and solved the problem of visible light captured synchronously by CCD for instantaneous incident photons.

Key words: scintillation optical fiber array, radiation imaging, GEANT, image reconstruction

前 言

本论文通过对 X 射线瞬态成像的实现原理，实验装置，实验方法及结果，计算机模拟研究以及相关成像问题的全面讨论，给出了瞬态 X 射线探测器在数字化实时成像方面的崭新研究成果。

论文介绍了当前数字化实时成像，尤其是 CT 和 DR 成像在医学，工业等各个领域中的广泛应用，全面地阐述了塑料闪烁光纤成像系统的物理学原理，论述了利用闪烁光纤阵列获取瞬态可见光，并通过 CCD 在计算机上获得数字影像，或是通过计算机模拟计算，图像重建的方法获得理论信息。论文中详细的讲述了闪烁光纤阵列成像探测器的设计实现，介绍了该系统作为瞬态成像探测器的实验应用，并结合实验结果论证了系统的可行性和先进性。

论文中详实的介绍了应用 GEANT 软件系统进行该探测器模拟的方法，利用 GEANT 系统建立了若干闪烁光纤成像探测器的模拟计算模型，计算并分析了以闪烁光纤作为探测单元的可行性和可靠性问题，得到了非常有意义的结果。

关于如何采用单根光纤实现 CT 和 DR 成像的问题，本文中讨论了一种具有重要应用价值的单根闪烁光纤成像方法。这种方法原理清晰，结构简单，成本低廉，对学校射线成像方面的教学具有一定的参考价值。

本文第一章主要介绍了数字化实时成像的基本知识和原理，以及当前的发展现状；在第二章中全面阐述了塑料闪烁光纤阵列成像探测器；第三章介绍了在射线成像领域中用来表征图像特性的几种参数；第四章介绍了图像重建的方法；第五章系统地分析研究了采用塑料闪烁光纤用于高能成像的可行性；第六章介绍了两种数字成像技术 CT 和 DR 的一些实验和模拟工作；第七章给出了研究结论，并提出进一步研究的若干问题和建议。

致 谢

本论文所有工作由我的导师阴泽杰教授亲自参与讨论、设计，并给予直接指导。他在研究工作中的教导和帮助使我获益良多，他活跃的学术思维、严谨的科研态度、忘我的工作精神都给我以极大的影响，必将鼓舞我在人生的道路上不断奋进。在此，我向他表示最衷心的感谢。

在五年的学习和研究工作中，我还得到了朱大鸣教授、张永明教授、吴孝义副教授和杜宪彬老师无私的关怀和帮助，得到了詹志锋、马奎等师兄师弟师妹们的热心支持。

在此，我谨向所有关心和爱护我的学习和成长的老师和同学们致以我崇高的敬意。

我还要向教育和培养我多年的中国科学技术大学表达我真诚的谢意，因为没有她“红专并进，理实交融”的良好学术氛围，就不可能有我成长的今天。

我要衷心感谢我的妻子杨莎莎。在我研究工作期间，是她不辞辛劳的照顾家庭，并同时给予精神上的支持与鼓励，使得我能按时完成学业。同时，我还要感谢我的岳父、岳母对我的支持！

最后，我要感谢生我养我的父母。没有他们的关爱和在精神和物质上的支持，就没有我成长的今天。

第一章 关于数字化X射线成像的研究

X射线成像的基本原理是利用射线对物体的穿透能力及其在穿透物体的过程中，不同密度的材料和不同物体结构对射线衰减程度的差异，使物体的内部结构在胶片，荧光屏，射线光电子器件或计算机等视频设备上形成影像。

1.1 X射线的性质及其衰减规律

1.1.1 X射线的性质

同一切电磁辐射和微观粒子一样，X射线也具有微粒和波动二相性[1]。在量子理论中，将X射线看成是一种量子或光子组成的粒子流，每个X光子具有的能量为：

$$E = h\upsilon = h\frac{c}{\lambda} \qquad (1.1)$$

式（1.1）中，E为X射线能量(电子伏)，h为普朗克常数，v为振动频率，c为光速，λ为波长。将有关常数代入后得：

$$E = \frac{12400}{\lambda}(eV) \qquad (1.2)$$

式(1.2)表明了X射线的波粒两重性。

作为一种电磁波，X射线在电磁波中占据介于紫外线和γ射线之间的波段，它的长波和短波也分别与真空紫外区和γ射线区重叠。一般来说，X射线波长范围从0.01nm到50nm，但5nm到50nm的长波段由于仪器条件限制，对其研究很少。X射线通常具有以下一般性质：

1）X射线沿直线以光速传播，其波长比可见光短，为不可见光线

2）X射线光子呈电中性，在电场和磁场中不发生偏转。

3）X射线和物质相互作用时，会产生光电效应，康普顿效应和电子对效应等物理现象并使物质电离，产生二次辐射和热效应等。

4）X射线具有透射、反射、折射、偏振、相干和不相干散射、衍射等光学性质。

1.1.2 X射线的产生

经典电动力学理论证实，作加速运动的电子产生电磁辐射[1]。作简谐振动的电子附近将产生同一频率的球面电磁波；而作为非简谐振动的电子，其运动可以分解为简谐分量和展开为傅立叶级数，而且在辐射过程中如果不提供任何外部能量，它的振幅将逐渐衰减，因此是

一种连续谱辐射而不是线性谱辐射。当运动的电子停（减）速时将释放出能量。设电子的初速度为v，停速时间为Δt，那么在电子停速时间Δt内释放的总能量为：

$$E=\frac{2e^2v^2}{3c^3\Delta t} \tag{1.3}$$

式(1.3)表明，电子的初速度越大，停速时间Δt越短，辐射的能量越大。

当电子从X射线管的阴极飞出时，在电场力的作用下向阳极方向移动，此时电子的速度不断加大，电场的势能转换为电子的动能，有：

$$U_e=\frac{1}{2}mv^2 \tag{1.4}$$

当高速电子到达阳极表面时，电子以不同的方式被突然制止，此时产生的电磁脉冲将会具有一系列连续的数值。X射线管中的电流强度通常是0.01A数量级，即每秒中有0.01C。电子的电荷为$e=1.6*10^{-9}C$，所以每秒内有$6*10^6$个电子达到阳极上。由于每个电子的停速时间各不相同，产生的电磁脉冲的波长显然也不同，而且数值是连续变化的。由此产生的波长连续变化的电磁脉冲就是连续X射线。

实验表明，连续X射线的总强度I随着X射线管内电流强度、电压和靶材料的原子序数的变化而变化。总强度（即阳极靶发出的连续射线的总能量）I是连续区的积分值，表示为：

$$I=\int_{\lambda_{min}}^{\infty} I(\lambda)d\lambda \tag{1.5}$$

式中，λ_{min}是连续谱的短波波长，$I(\lambda)$为按波长分布的射线强度表达式。对于波长小于5$\mathring{A}$的连续射线有以下近似计算公式：

$$I(\lambda)=\frac{cZi}{\lambda^2}\left(\frac{1}{\lambda_{min}}-\frac{1}{\lambda}\right) \tag{1.6}$$

利用式(1.5)，(1.6)，通过积分得到连续X射线的总强度$I_{总}$：

$$I_{总}=kZiU^m \tag{1.7}$$

其中，k值为常数（约为$1.1\text{-}1.4*10^{-9}$），Z为靶材料原子序数，m约为2，U单位为伏。若将连续谱X射线以单色射线等效，其等效单色射线能量大约等于射线谱中最短波长所对应能量的2/3。

在X射线管中，由炽热的灯丝加热射线管阴极提供电子，外部施加高压电场来驱使电子向阳极运动，当电子在靶面上突然停止就导致了X射线的产生。X射线管所发射的X光子能

量通常在几十到几百千电子伏范围内。当射线检测的能量达到 1MeV 或更高时，X 射线管就不能满足要求了。最高的高能工业射线源是 1939 年提出的 1MeV 共振变压器式 X 射线机，后来又发展了静电起电式 X 射线机、电子感应加速器以及电子直线加速器。目前在工业检测中使用的最多的是电子直线加速器，它采用谐振波导来加速电子轰击靶极，可产生范围在 2-15MeV 或更高能量的射线。

1.1.3 射线与物质的相互作用

当 X 射线为单色时，假设贯穿厚度为 x 的物体后 X 射线的强度为 I，厚度增加为 x+dx 时，射线强度变为 I-dI，根据实验结果，薄层 dx 的增加与射线强度的减小率-dI/I 成正比，即：$-dI/I=\mu dx$，其中 μ 为物质对射线的线性吸收系数。

将上式积分并简化得：

$$\int -\frac{dx}{I} = \int \mu dx$$

$$-\ln I = \mu x + C$$

则 $$I = e^{-\mu x} e^{-c}$$

令 $$e^{-c} = I_0$$

得 $$I = I_0 e^{-\mu x} \qquad (1.8)$$

其中，I_0 为贯穿物体前得射线强度。式(1.8)表明射线穿透物质后其强度按指数方式衰减。

通常，μ 值随着射线能量的增加而减小，随物质原子序数的增大而增大，随物质密度的增大而增大。这也说明了能量越高的射线穿透能力越强，而密度越大的物体越难穿透。当射线强度衰减为入射前的一半时，穿过物体的厚度被定义为这种物质的半值层 HVL(half-value layer)。

当 X 射线谱不是单色，而是连续谱时，射线中各个波长部分都以不同的衰减系数进行衰减，结果总的射线剂量以一种较复杂的方式被吸收，但随着物质厚度的增加，物质对 X 射线的硬化作用趋向于使 X 射线接近单色，当射线穿透物质的 2-3 个半值层时，就可以认为射线是单色的了，并有一个等效的 μ 值。在上一节已有说明，连续 X 射线的等效单色射线能量大约等于射线谱中最短波长对应能量的 2/3 倍。因此，取该射线所等效的单色 X 射线所对应的 μ 值即可。

X射线不带电，在物质中没有直接的电离和激发效应，不能直接被探测到。但是X光子在物质中会产生光电效应，康普顿散射以及电子对效应，通过这些作用所产生的次级电子再引起物质的电离和激发，从而可以被探测到。下面分别介绍这三种作用：

（1）光电效应

光子通过物质时和物质原子相互作用，光子被原子吸收后发射轨道电子的现象，称为光电效应。

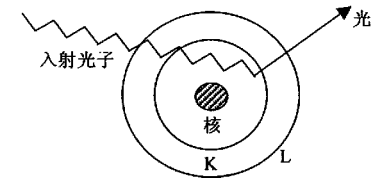


图1.1　X射线与物质相互作用的光电效应

当 $h\nu << m_e c^2$ 而 $h\nu > B_K$ 时（B_K 是K层电子的结合能），对于K层电子，每个原子的光电截面为：

$$\sigma_{KPh} = 2^{5/2}\alpha^4\left(\frac{m_e c^2}{h\nu}\right)^{7/2} Z^5 \sigma_{th} \qquad (1.9)$$

当 $h\nu >> m_e c^2$ 时：

$$\sigma_{KPh} = 1.5\alpha^4 \frac{m_e c^2}{h\nu} Z^5 \sigma_{th} \qquad (1.10)$$

上式中：$\alpha = e^2/hc = 1/137$ 是精细结构常数，σ_{cth} 是每个电子的汤姆逊散射截面，其值为：

$$\sigma_{cth} \approx \frac{8}{3}\pi\left(\frac{e^2}{m_e c^2}\right)^2 = \frac{8}{3}\pi r_e^2 = 6.651\times10^{-25} cm^2 \qquad (1.11)$$

发生光电效应时入射光子必须用一部分能量来克服电子在原子中的结合能，L和M层上产生光电效应的几率远小于K层，原子光电效应总截面大约为 σ_K 的5/4倍。

（2）康普顿散射

康普顿散射发生在入射光子与物质原子核外的轨道电子之间。入射的X光子能量较低（$h\nu << m_e c^2$）时，只改变运动方向而不损失能量；其能量接近或超过 $m_e c^2$ 时的散射称为康普顿散射。康普顿散射可以下图来表示：

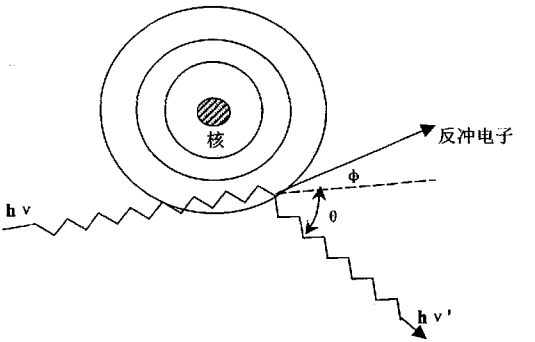


图 1.2 X 射线与物质相互作用的康普顿散射

根据相对论的能量和动量公式以及康普顿散射的能量和动量守恒，可以写出以下方程：

$$\left.\begin{aligned} h\nu &= m_e c^2(\gamma-1)+h\nu' \\ \frac{h\nu}{c} &= m_e c\beta\gamma\cos\phi+\frac{h\nu'}{c}\cos\theta \\ 0 &= m_e c\beta\gamma\sin\phi-\frac{h\nu'}{c}\cos\theta \end{aligned}\right\} \qquad (1.12)$$

式中 $\beta=v/c$，v 为反冲电子速度，$\gamma=(1-\beta^2)^{-1/2}$。由上面方程组可以得到如下关系：

$$h\nu'=\frac{h\nu}{1+\frac{h\nu}{m_e c^2}(1-\cos\theta)} \qquad (1.13)$$

$$E_{ce}=h\nu-h\nu'=\frac{(h\nu)^2(1-\cos\theta)}{m_e c^2+h\nu(1-\cos\theta)} \qquad (1.14)$$

$$ctg\phi=(1+\frac{h\nu}{m_e c^2})tg\frac{\theta}{2} \qquad (1.15)$$

康普顿散射截面可以用下式表达：

$$\sigma_{ce}=2\pi r_e^2\{\frac{1-\alpha}{\alpha^2}[\frac{2(1+\alpha)}{1+2\alpha}-\frac{1}{\alpha}\ln(1+2\alpha)]+\frac{1}{2\alpha}\ln(1+2\alpha)-\frac{1+3\alpha}{(1+2\alpha)^2}\} \qquad (1.16)$$

上式中 α 为 $h\nu/m_e c^2$。当 $\alpha<<1$ 时，有：

$$\sigma_{ce}=\sigma_{cth}(1-2\alpha+\frac{26}{5}\alpha^2+\ldots)\approx\sigma_{cth}(1-2\alpha) \qquad (1.17)$$

此时就是汤姆逊散射的情形 $\sigma_{ce}=\sigma_{cth}$。

（3）电子对效应

在入射光子能量大于1.02Mev时，它在物质中经过原子核附近时，与原子核发生电磁相互作用，入射光子消失，同时产生一个电子和一个正电子（即电子对），称为电子对效应。电子对效应是在入射光子和原子核的共同参与下完成的，作用过程满足能量和动量的守恒。X光子在一个原子核上发生电子对效应的总截面可以由理论计算得到：

当 $2m_ec^2<h\nu<137m_ec^2Z^{-1/3}$ 时，

$$\sigma_P=\frac{Z(Z+1)r_e^2}{137}[\frac{28}{9}\ln(\frac{2h\nu}{m_ec^2})-\frac{218}{27}] \tag{1.18}$$

当 $h\nu>>137\ m_ec^2Z^{-1/3}$ 时，

$$\sigma_P=\frac{Z(Z+1)r_e^2}{137}[\frac{28}{9}\ln(\frac{183}{Z^{1/3}})-\frac{2}{27}] \tag{1.19}$$

X射线与物质相互作用的三种效应的发生几率（与作用截面有关）在不同的情况下是有很大不同的。简单地概括起来就是，光电效应在低能高Z区占优势，康普顿散射在中能低Z区占优势，电子对效应在高能高Z区占优势，这可以用图1.3来说明：

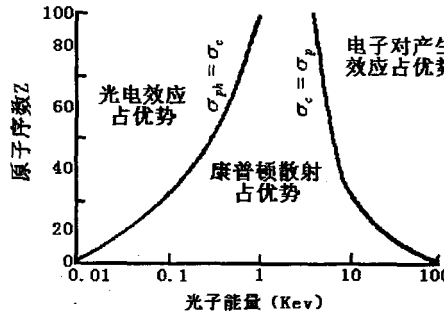


图1.3 按照光子能量和原子序数表示的三种相互作用占优势的区域示意图

1.2 数字化X射线成像

1.2.1 X射线探测方法[1]

探测X射线的最常用的方法之一是采用闪烁探测器。

闪烁探测器是目前应用最广的核探测器之一，它是利用某些物质在核辐射的作用下会发光的特性探测核辐射的，这些物质就是闪烁体。利用光电器件将微弱的闪烁光转变成光电子，再经过多次倍增放大，然后输出的是电脉冲。

闪烁探测器的工作原理可以这样描述：粒子进入闪烁体，发生光电效应、康普顿散射或电子对效应，把能量传给电子，这些电子最终通过电离或激发作用将能量沉积在晶格中，使原子或分子激发，受激原子在退激过程中发光，光子穿过闪烁体和光导，在光电倍增管中经过若干级倍增过程后，输出一个电脉冲信号。通过对电脉冲信号的探测，就可以研究入射粒子的有关物理学特性。

闪烁探测器的性能与闪烁体有很大关系。闪烁材料可以分成无机闪烁体和有机闪烁体两大类：典型的无机闪烁体有NaI(Tl)晶体，CsI(Tl)晶体，ZnS(Ag)晶体，BGO，BaF_2晶体，玻璃闪烁体和气体闪烁体等。无机闪烁体具有较高的探测效率，但空间分辨率达不到很高，并且容易潮解，吸收空气中的水分发黄变质而不能使用，不适合长时间使用；常用的有机闪烁体有塑料闪烁体，蒽晶体和液体闪烁体等。在高能下，闪烁光纤具有比碘化钠，碘化铯等无机闪烁体更好空间分辨率特性。

1.2.2 数字化X射线成像概述

“数字化X射线成像”实质就是在闪烁探测器的基础上，利用电子学方法获取探测器得到的闪烁光，从而得到所成的图像。即利用电子学手段对图像进行获取，处理，存储，显示和传输等。

数字化X射线成像的第一步是图像获取。传统的图像获取设备是采用荧光屏和胶片。这种技术以及相关的胶片处理技术发展到今天已经是非常完善，一些专门的荧光屏和胶片已被研制出来。例如乳腺X射线照相术等。但随着对图像质量要求的越来越高，这种传统的方法逐渐显示出很大的局限性，例如在动态范围，间隔尺度以及对比度上。而数字化成像则具有高探测效率，高动态范围以及高对比度的优点，这使它在成像技术中有着巨大的潜力。在图像处理方面也得到革新，例如利用计算机能够进行诊断，图像融合，双能照射以及数字减影等技术。

如今数字化成像一般有几种方法，这其中一些正处于研究阶段，一些则已经发展的比较成熟，进入市场。一般来说，探测器是主要研究对象，这其中有固定的，缝扫描的，可移动式转换屏。探测方式一般有直接探测和间接探测两种：直接探测例如采用无定型的硒或者CdZnTe探测；间接探测则是利用闪烁体把X光转换成可见光后采用可见光探测器，例如CCD等。

1.2.3 数字化X射线成像系统结构

本文研究的射线数字成像系统是一种综合性、开放式的实时成像系统。射线源发出的X射线对试件进行透射，通过射线转换体，实现入射射线向弱可见光的高灵敏度转换，并成像在专用的科学级CCD相机上，然后通过图像采集和处理模块，得到在计算机上显示的数字图像。图1.4是一个基本的成像系统结构图。在本文所涉及的成像系统中，射线转换体和CCD相机是主要的成像器件，也是决定原始图像质量的主要因素。决定射线转换体效率的关键是闪烁体。本文涉及的闪烁体材料为塑料闪烁光纤。

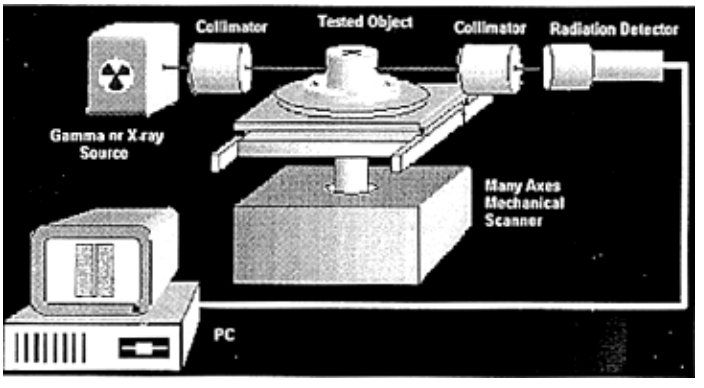


图1.4 典型的数字化成像系统结构图

随着CCD技术的发展，高灵敏度，高动态范围的科学级CCD相机已广泛应用此系统。由于它的暗电流和读出噪声极小，可实现长时间电荷积累，并且具有高的灵敏度和量子效率，适合于微光检测，能完全满足本系统的图像采集要求。与普通相机相比，它具有以下优点：（1）高灵敏度和高量子效率（2）低噪声（3）高动态范围（4）低暗电流（5）线性度好且输出基线稳定（6）高空间分辨率（7）灵活的扫描模式，可用于对微光信号检测和微光成像。在射线数字图像检测、天文观测、生物医学工程、武器装备、水下摄影等多种技术领域得到了广泛的应用。

射线转换体利用荧光物质或晶体将X射线转换成为可见光，荧光和闪烁属于同一种物理过程，射线光子激励荧光物质后可产生小于10^{-8}秒的可见光持续，而闪烁晶体对射线的转

换是单个射线光子和诱发粒子激发的瞬时闪光脉冲。转换体特性由其效率、发光光谱、不清晰度和灰度系数所决定。理想闪烁体应具有以下性质：（1）将射线粒子动能转变成闪烁光子的效能（光产额）高；（2）入射射线粒子损耗的能量与产生的闪烁光子数应当是很好的线性关系，成线性关系的能量范围愈大愈好；（3）为了能收集到入射粒子产生的可见光，闪烁体介质发射光谱与吸收光谱不应该重叠；（4）入射粒子产生的闪光持续时间（发光时间系数）要尽量短，以便于能产生快的输出信号；（5）良好的加工性能、稳定性及合适的折射率等。

射线源在成像系统中也是极为重要的一个因素。传统的热阴极X射线在实时成像中得到了广泛的应用。半波整流、全波整流和恒压机在X射线的输出波形上有一定的差别，当成像周期小于X射线的波形变化频率时，这种波形变化的影响就应该加以考虑。在使用直线加速器的场合，高能X射线是采用谐振波导来加速电子轰击靶极，射线的脉冲宽度一般是3-4μs，频率是60-420Hz，一般我们称这种射线源为瞬态射线源。在实时成像过程中，必须保护检测元件不受射频噪声的干扰和高强度射线的直接辐照。

1.3 CT与DR成像技术

CT与DR成像是数字化成像领域中不同的两种实时成像方式，其应用都极为广泛，涉及医学、生物、工业、安全等各个领域。随着被测对象愈来愈广泛，对成像系统要求越来越高，一些高分辨率，高对比度，高灵敏度，实时性强的CT和DR成像探测器应运而生，这些成像探测器的出现使得传统的闪烁体已不能满足要求，具有更小单元象素的闪烁体成为高精度成像探测器的追求目标。

1.3.1 CT的概述[2]

CT，即计算机断层成像，自从20世纪70年代至今，已经在医学以及许多科技领域中得到广泛的应用。早在1917年，J. Radon就提出了CT图像重建的基本数学理论，当时并没有引起人们足够的重视。20世纪70年代Cormak和Hounsfield几乎同时推出第一台医用CT机，并于1979年二人同时获得诺贝尔医学、生理学奖，1985年，G. T. Herman在他的专著中系统的阐述了CT的理论基础。20世纪80年代初期，对三维物体重建的研究取得了突破性的进展。

目前，世界上一些科技发达的国家都在竞相研究CT技术，在80年代末期已经完成了第五代CT的研制工作，已发展出近千种CT产品，适用于航空航天、材料学与工程、核科学与工程、生物医学与工程、物理化学、控制工程、计算机科学与工程、应用数学、微电

子学、机械工业、冶金、石油化工、建筑等行业。

CT经过几十年的发展，已日趋成熟，其卓越的性能被誉为二十世纪后期最伟大的科技成果。其检测能力是以往任何检测设备不可比拟的，它不受被检物体的材料、形状、表面状况影响，能给出被检物体的二，三维直观图像。CT系统按照能量可分为低能(X射线能量小于450keV)和高能(能量大于1MeV)两类，按照扫描系统的结构可分为线阵探测器CT系统和面阵探测器CT系统。大部分CT系统均采用以滤波反投影算法为基础的重建算法，对一些特殊构件的检测，也有采用有限投影数据的迭代算法。

可以肯定，随着CT理论研究与计算技术的发展，CT必将对推进国民经济和国防现代化建设做出越来越大的贡献。

1.3.1.1 CT的基本工作原理[2]

X射线CT成像的基本原理是X射线穿过物体后构成的二维投影为传感器所接收，然后利用计算机进行图像重建来获得一个完整的断层图像。X射线CT的一个简单物理模型如下：

设$f(x)$是点x处物质的衰减系数，I_0为X射线的初始强度，I_1为X射线通过物体后的强度，则有：

$$I_1 = I_0 \exp\{-\int_L f(x)dx\} \qquad (1.20)$$

对(1.20)式两边取对数有：

$$\ln\left(\frac{I_0}{I_1}\right) = \int_L f(x)dx \qquad (1.21)$$

其中I_0和I_1都是可观测量，CT成像的过程就是通过测量足够多的数据$\ln\left(\frac{I_0}{I_1}\right)$估计$f(x)$的过程。现行衰减系数$f(x)$反映了物体空间物质分布的密度，因此他在空间上就形成了物体的图像。

所谓投影其实是一种变换，它将一个N维函数某一特定方向的积分变成一个(N-1)维函数。如图（1.5）中$P_\theta(x') = \int_L f(x,y)dl$就是X射线沿着与Y轴成$\theta$角的$Y'$轴的投影。

所谓反投影是指下属意义下的投影之逆：用某视角θ下的投影数据对所有的u_1方向

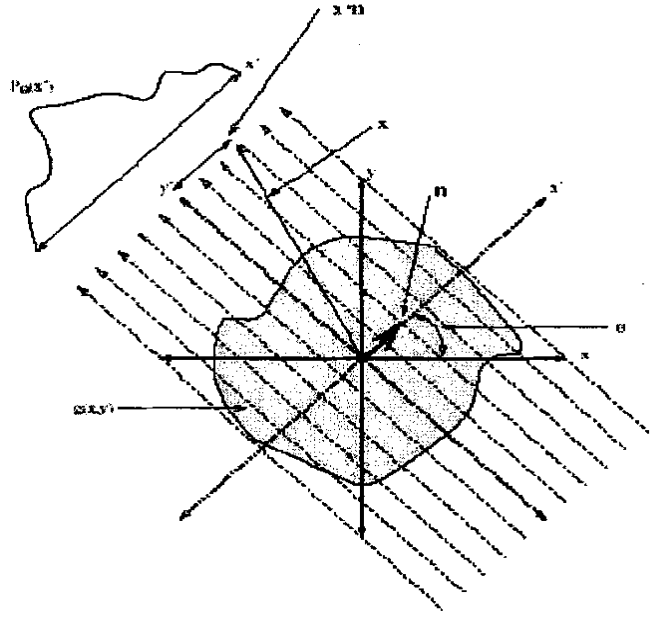


图 1.5 沿 θ 角方向投影的二维分布

的像素点贡献相同的值，从而得到一个二维分布 $g(u_1,u_2)=p_{u_1}(u_2)$，并赋以密度的量纲。其数学表达为：

$$x_k=\frac{1}{n_p}\sum_{i=1}^{n_p}P_{k,i} \tag{1.22}$$

CT 的目的是得到在物质内部占有确切位置的物质特性的有关信息。一张 CT 得到的断层扫描图是一个切片的图像，是从一个指定方向截取的物质内一个解剖平面的显示。这种图像的重现使我们能发现并显示出所观察部分的精确形状。普通的 X 射线照相断层摄影目前已得到广泛的应用，而通用放射性同位素断层扫描的应用还不太普遍。

1.3.1.2 CT 设备的构造及分类

CT 系统主要由射线源、成像系统（获取投影）、机械扫描系统和软件处理系统等四大部分组成。图 1.6 为一个典型 CT 系统的组成构造。低能 X 射线 CT 系统一般采用 X 射线管作为射线源，产生低于 450keV 的 X 射线；高能 X 射线 CT 系统一般采用电子直线加速器作为 X 射线源，产生 1MeV 以上的 X 射线。

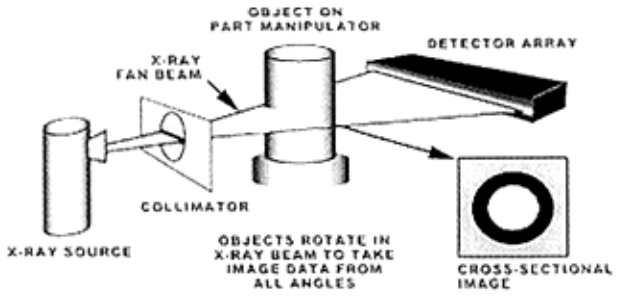


图 1.6 典型 CT 系统的组成示意图

根据所采用成像系统结构的不同可分为基于线阵探测器的CT系统和基于面阵探测器的CT系统。

基于线阵探测器的CT扫描系统的结构为射线源+前准直器+CT转台+后准直器+线阵探测器+计算机控制重建系统。这类设备对物体一次扫描可重建被测物体的一个一维断面图像，在进行多次扫描并重建出相邻区域多个断面的二维CT图像后，可堆积出该区域的三维灰度图像。线阵探测器是由一系列单个探测器单元组成，每个探测器单元由闪烁体、光电二极管和放大器组成，加上数据采集单元构成成像系统。这种系统由于较好的消除了散射的影响，重建出的CT图像非常清晰。当前英国、法国、德国、俄罗斯、印度、澳大利亚和中国均有低能CT产品。

目前国内还没有研制开发出基于线阵探测器的、以加速器为射线源的高能CT系统。

基于线阵探测器的CT系统有其许多优点，但线阵探测器的个数一般在10^3数量级，一次扫描仅能获取一个切片的一维投影数据，重建一个切片的CT图像，为此基于DR系统二维闪烁晶体＋面阵探测器的高能X射线CT技术受到人们的关注。这种系统采用锥束射线，投影获取系统采用闪烁晶体＋面阵探测器结构，闪烁晶体将透射X射线转换为可见光，由镜头耦合到面阵CCD探测器，一次扫描可获得物体一个区域的二维投影图像，故其扫描速度快，射线利用率高。因加速器的运行费用高，故这一特点对高能X射线非常重要。目前面阵CCD探测器的像素尺寸在微米数量级，像素个数在10^6数量级，因此获得的图像空间分辨率高。高能X射线穿透物体时，散射是主要的衰减源，然而对于基于面阵探测器的高能X射线CT系统，由于无法对闪烁晶体和每个探测器单元进行后准直，散射的结果降低了射线图像的空间分辨率，因此散射校正成为一个重要的工作。

根据CT成像技术用途的不同，可分为医用CT成像技术和工业CT成像技术。两者有许多相同之处：他们都采用电磁波作为信息载体，都以样品线衰减系数为检测对象，用点源

和相同的扫描原理等。因此原则上说，医用CT也可应用于某些工业检测。在检测组分原子序数不高，密度低、尺度适当的条件下，可以代用。

1.3.2 实时DR成像技术

实时DR成像(Digital Real Time Radiography)目前被广泛应用在医疗、军事、航天以及大型工业设备制造行业的检测中。它能以高的透射灵敏度和高的空间分辨率对被测物体做实时成像的透视检测。若在X射线DR检测基础上，辅助以CT检测的功能，不仅能获得构件的DR图像，而且能给出构件的断层图像，显示缺陷的大小、形态及空间位置等信息，对于准确判读构件内部的缺陷是非常有意义的。

1.3.2.1 X射线DR检测系统的组成

射线数字成像（DR）技术是以射线转换和可见光图像数字采集设备代替胶片获取射线图像的技术[3]。X射线DR系统一般由X射线源、射线转换屏、高性能的CCD相机和计算机控制系统组成。图1.7是一个典型的X射线DR系统组成图。DR检测过程为：X射线穿过试件到达转换屏（闪烁屏）后，一部分射线穿透转换直射前方，一部分射线被转换屏吸收，将透射线能量转换为可见光，即将射线透视图像转变为可见光图像；可见光图像经平面反射镜反射改变方向后，被CCD相机采样为数字图像，并通过控制器、适配器传送给计算机，由计算机图像进行处理、输出。

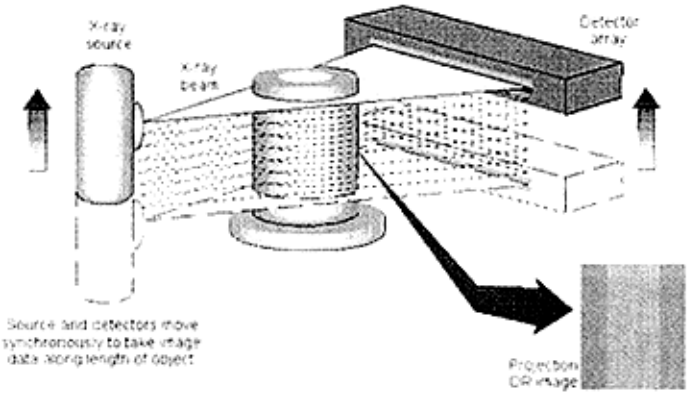


图1.7 典型的X射线DR系统组成图

高能X射线能量高、穿透能力强。如果将CCD相机直接耦合在转换屏后面采样可见光图像，则未被转换屏吸收的高能X射线将直接照射在CCD相机上，一方面会使CCD相机的输出图像噪声增大，另一方面高能射线长时间直接照射CCD相机，会使其损坏。因此，高能X射线DR系统中一般都要用反射镜使转换屏上的可见光图像传播方向改变90°，并且CCD相

机被屏蔽在几十毫米厚的铅室内。有时为了更好的保护CCD相机，还有使用两块反射镜的高能X射线DR系统。

1.3.2.2 X射线DR检测系统的分类

从广义上，DR分为直接数字X线摄影(DDR)和间接数字X线摄影(IDR)两大类。从狭义上讲，DR就是指直接数字X线摄影。IDR(indirect digital radiography)是在X线电视系统的基础上，利用计算机数字化处理，经过采样、A/D转换后将模拟信号转变为数字信号。其成像方式有影像增强器与电视系统(I.I-TV)和电荷耦合器件与电视系统(CCD-TV)两种。DDR (direct digital radiography) 指采用一维或二维X线探测器直接将X线影像转换为数字信号进行数字化处理的成像方法。

1.4 数字化成像的基本原理

从以上对CT和DR的研究可以得出，数字化成像基本原理是利用闪烁体探测器把X射线经被测物体转变为可见光，利用电荷耦合器件CCD或光电倍增管等设备把光信号转变成电信号进入计算机视频采集或视频设备，并采用某种图像重建算法实时得到被测物体图像。因此选择合适的射线探测器、电荷耦合器件，视频采集、图像重建算法是建立数字化成像系统的几个关键步骤。以下章节我们将对这些关键环节做进一步的研究。

1.5 数字化成像的意义

众多传统的成像方法虽然也仍在很多领域得到运用，但是已不能够满足现有的应用和研究的需要，最根本的困难就是空间分辨率不高，并无法进行实时处理。而数字化成像技术目前已逐渐在国民生产生活许多领域得到应用，尤其在国防，航空，医疗等领域，并且日益发挥着不可替代的作用。数字化成像技术具有很强的实时性，而且由于采用数字化图像处理设备，提高了系统信噪比，因而在图像质量方面也具有极大的优势，这对于工业上物体缺陷检测，医疗上的病理检测等方面都具有极大的意义。因此可以说数字化成像技术适应了当今数字化时代的发展，在射线成像技术发展过程中具有划时代的意义。而随着科学的进步，计算机模拟技术、数据采集技术、计算机图像重建及处理技术以及信号的传输技术等方面的发展已经为数字化成像技术提供了保障。正是在这样的前提下我们提出了基于塑料闪烁光纤阵列的成像方法，以期在高能数字成像方法的研究上有所突破。

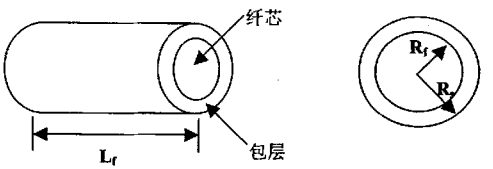
第二章 塑料闪烁光纤成像探测器

作为成像探测器中的一部分，闪烁探测器是决定成像系统质量最关键的因素。它对整个探测器的吸收效率，灵敏度，信噪比以及最后所成图像的空间分辨率，对比度等都起着决定性的作用。因此，选择闪烁探测器是成像系统设计中最重要的环节，其标准是根据系统总体要求，包括系统实时性要求、图像分辨率要求、曝光时间、探测效率、射线源能量等综合因素。根据我们的研究目的是对低能和高能瞬态 X 射线辐射成像的研究，因此在众多射线闪烁体中我们选择了塑料闪烁光纤。

2.1 关于塑料闪烁光纤

2.1.1 塑料闪烁光纤的一般结构

闪烁光纤一般由纤芯和包层构成，如图 2.1 所示。



a. 光纤结构　　b. 光纤断面

图 2.1 光纤结构示意图

图中 R_f 为光纤纤芯半径，R_c 为光纤外半径（包括包层厚度），L_f 为光纤长度。作为产品，一般在闪烁光纤包层外部还有一层吸收层 EMA(Extra Mural Absorber)，主要作用是吸收相互作用产生的一些次级粒子，以减小闪烁光纤阵列中光纤之间的相互串扰。

2.1.2 塑料闪烁光纤的材料及传输特性

研究中我们所选择的闪烁光纤型号是 BCF-20，其纤芯(core)材料为聚苯乙烯（单体分子式为 $C_6H_5CH=CH_2$），包层(clad)材料是聚乙烯醇（单体分子式为 $CH_2=CHOH$）[7]。聚苯乙烯的密度为 1.05g/cm^3，折射率为 1.6；聚乙烯醇的密度为 1.26g/cm^3，折射率为 1.49。一般闪烁光纤包层的厚度为纤芯直径的 3%—5%。就 BCF-20 而言，输出的可见光子的波长峰值是 492nm，每 1MeV 的能量沉积产生 8000 个可见光子，衰减时间是 2.7ns。根据纤芯和包层的折射率，可计算出全反射临界角 θ_c。（若光波入射角大于 θ_c，则光波就可沿着光纤传输）。根据折射

定律，即斯涅耳(Snell)定律：（其中：θ_1为入射角、θ_2为折射角）

$$\frac{sin\theta_1}{sin\theta_2}=\frac{n_2}{n_1}=\frac{n_g}{n_c} \tag{2.1}$$

如果 $n_1>n_2$，表明光波是由光密介质入射向光疏介质，则在某种情况下会发生全反射，其临界角为：

$$\theta_c=\arcsin(\frac{n_g}{n_c})=42.8^\circ \tag{2.2}$$

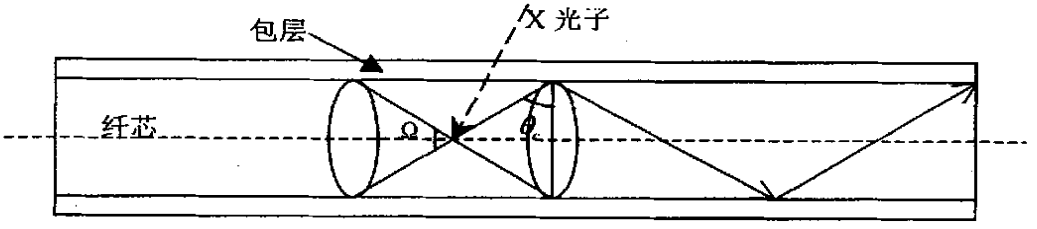


图 2.2 光纤中光的传输

假设在光纤纤芯中闪烁光是各向同性的，即从发生作用的位置，向前和向后传送的光是相同的，则可算出能被收集到的沿着光纤传输的部分光（参见图 2.2）。这部分可定义为闪烁光所有可能发生全反射形成的圆锥形包含部分，其立体角Ω为：

$$\Omega=2\pi(1-sin\theta_c) \tag{2.3}$$

则沿光纤传输的部分光占总光量的百分数为：

$$\frac{\Omega}{4\pi}=3.44\% \tag{2.4}$$

当在光纤纤芯中X射线的作用发生时，获得的全部光能的3.44%被传送至连接到光探测器的光纤终端。在相反方向，相同数量的光也同样通过光纤纤芯进行传输。以上讨论仅适用于通过光纤轴心平面(称子午面)的那部分光的传播，即沿光纤轴线方向传输。若光线不通过光纤的轴心平面，则称其为斜射线。当闪烁光在远离光纤轴心线处产生时，还会有一些额外数量的非子午线光波(斜射光传送模式)存在，其约是子午线光波的60%。

2.1.3 塑料闪烁光纤的闪烁探测原理

根据闪烁光纤特性可以看出作为闪烁探测器，它既是射线探测的灵敏元件，又是光信号传输的元件。如图2.3所示。

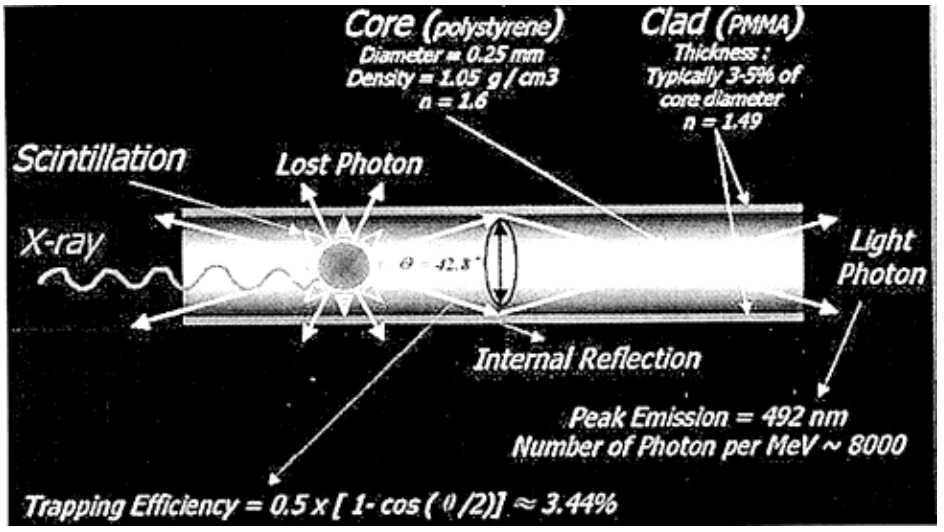
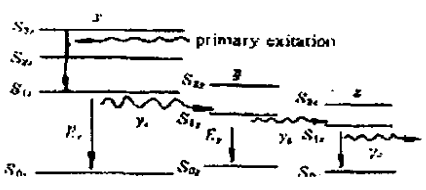


图 2.3 闪烁光纤的材料及特性

纤芯是闪烁材料构成的光纤能适应 E>5kev 的 X 射线及其它射线的辐照探测。在我们所涉及测量的能量范围内，X 射线在塑料闪烁光纤中的主要作用机制是康普顿效应和电子对效应，这是由于闪烁光纤本身是一种有机物，具有较低的原子序数 Z，但一般的闪烁光纤里都加入了一些高原子序数，低激发态的闪烁体，其作用有两点：加入原子序数高的闪烁体可以增加闪烁光纤的吸收效率，改善闪烁光纤探测灵敏度；加入低激发态的闪烁体，起着分离闪烁光纤对于可见光的吸收谱和发射谱的作用。目前的普遍认识是：电磁辐射作用产生了原子激发，材料中原子退激发时会发光。闪烁光纤苯环中 π 电子受激发，跃迁到高能级，而后又自发跃迁到低能级或基态，并发出一个光子，荧光光子的波长有一定范围，通常发射谱和吸收谱有部分重叠，造成荧光的自吸收。因此在实际应用中，要在闪烁物质(x)中加入杂质(y, z)，它们一般也是闪烁物质，但与 x 相比具有较低的激发态。加入杂质后，x 发出的荧光光子 γ 会被第一种杂质吸收，释放出具有更大波长的光子 γ 1，γ 1 还会发生和第二种杂质相同的过程，把荧光变到波长更大的范围。最终被探测到的是光子 γ 1 和 γ 2。因此这里的 y, z 是起着分离吸收谱和发射谱的作用，我们称之为移波剂[1]。以上过程如图 2.4 所示。



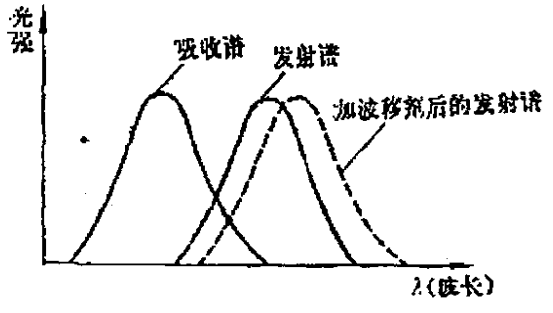


图 2.4 闪烁光纤产生可见光的原理示意图

2.1.4 塑料闪烁光纤在射线成像方面的优势

同传统的射线成像探测器相比，闪烁光纤具有以下优点：

1）高空间分辨率

较传统的闪烁屏而言，闪烁光纤最大的优点之一在于能够根据需要灵活的伸长或缩短光纤的长度来改变探测器效率，从而可以尽量提高图像信噪比，但同时对图像的分辨率影响不大，这主要是由于光纤自身的横向光传导的特性决定的。

2）衰减长度

闪烁光纤具有较长的荧光传输衰减长度，光纤束所成的二维图像，可以在很长的距离上传播，同时由于闪烁光纤的柔韧性，可延伸到空间的某些大探测器不易安放的狭小地方，扩大了探测范围。

3）光纤大小

对于无机闪烁体，例如CdWO4，它的可达到的最小象素点是固定的，而闪烁光纤则较为灵活，为了分辨率的需要，我们可以选择直径非常小的闪烁光纤作为成像探测器，如今最小的直径可以到达10um。

4）快速响应

闪烁光纤的延迟时间大约只有2－3ns，可以将其运用于射线瞬态成像领域。

5）光学匹配

闪烁光纤输出光子的光谱与大部分感光探测器如PMT，CCD等的感光响应曲线相匹配。

6）线性度

塑料闪烁光纤的产额，即在X或γ射线入射下，光子在光纤中单位能量损失所产生的可见光子数量是线性的，而且在不同的入射强度下也是保持线性的。

7）光导灵活性

通过改变光纤的几何形状，可以非常容易的改变光输出方向。这实际上是由于光纤本身也是起一个光导的作用。

8）电磁免疫性

闪烁光纤探测器能够在较强的电磁场条件下稳定工作。

9）易保存

传统的一些无机闪烁体易与周围的环境发生作用，如遇空气潮解，而闪烁光纤则不存在

类似的问题，它可以放在任何开放的环境和室温当中。

10）便于携带

闪烁光纤束不易碎，而且无辐射，因此便于携带。

11）装配简单

闪烁光纤阵列的装配非常方便，无需高档工具。

12）低成本

与所有的数字成像探测器相比，塑料闪烁光纤阵列的成本是较低的，因此很容易得到普及。

2.2 塑料闪烁光纤成像探测器的系统设计

根据以上闪烁光纤自身特性和闪烁探测原理，我们可以利用这些特性设计一种在低能和高能下，具有实时性，系统空间分辨率和对比度满足指定要求的X射线瞬态成像探测器。

2.2.1 系统设计

闪烁光纤阵列成像探测器系统主要包括二维闪烁光纤阵列探测器、CCD 同步接收与输出、前置放大、视频放大及数据采集、定点采集控制和计算机、数据处理等部分。其结构如图 2.5 所示

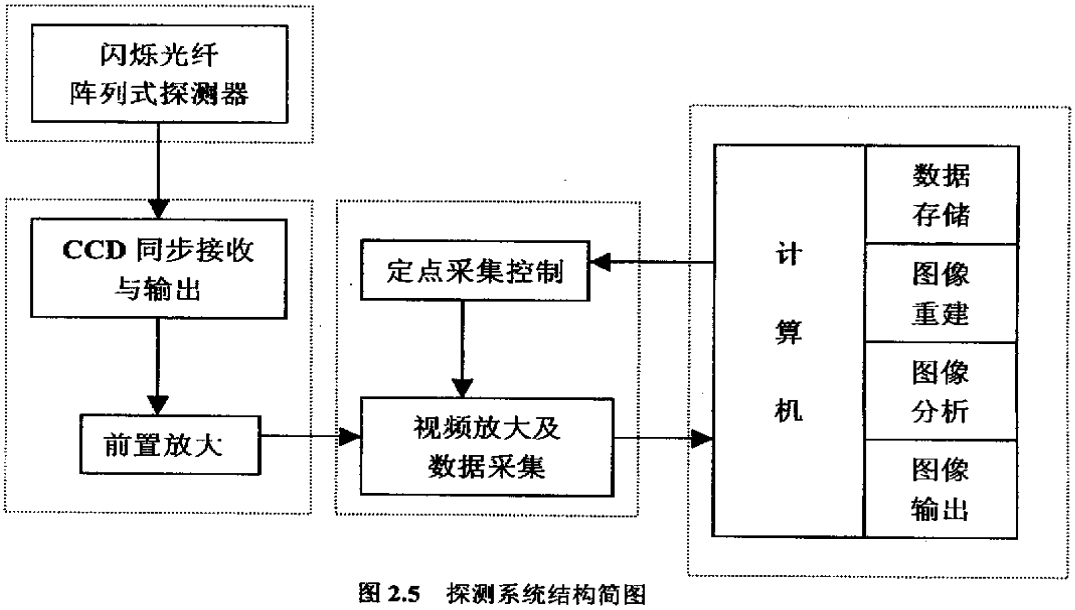


图 2.5　探测系统结构简图

闪烁光纤阵列探测器与X射线作用产生可见光，光线由光纤传输到CCD（电荷耦合器件）摄像头。为了满足在瞬态发出X射线时，CCD能够在瞬间采集到视频信号，必须设计一个CCD同步信号，使发生可见光的同时，CCD处于有效视频采集状态。采集到可见光后，由摄像头引出视频信号经过前置放大后再进行视频放大，经过放大的视频信号在定点采集逻辑

的控制下完成数据采集，并以数据文件的形式存入计算机硬盘。在计算机系统里，可以对数据进行处理，从而实现图像重建、图像分析和图像输出。

2.2.2 闪烁光纤阵列式探测器

探测器部分是整个探测系统的核心所在，它的结构可以如图 2.6 所描述。

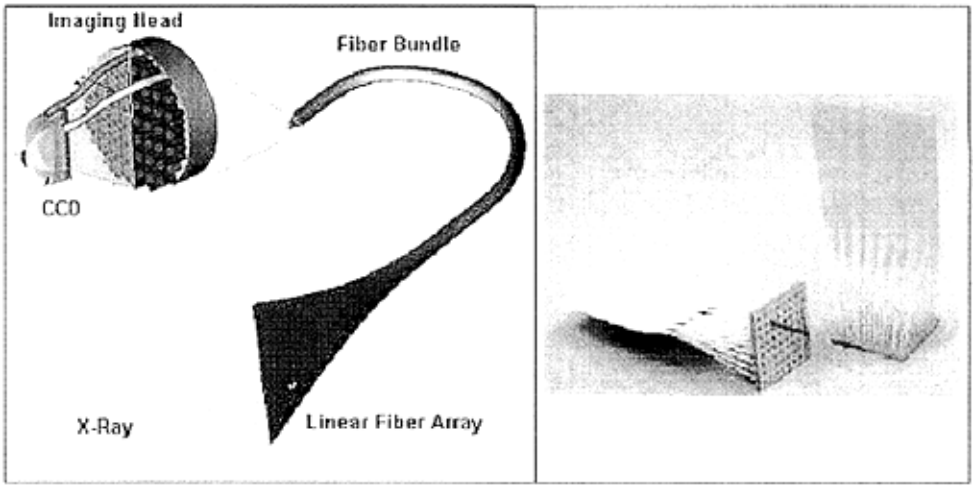


图 2.6　闪烁光纤阵列式探测器结构示意图

闪烁光纤阵列由 20×35 根闪烁光纤排列成二维阵列构成，闪烁光纤的直径为 1mm，断面为圆形，光纤长度约 10cm，如图 2.7 所示。闪烁光纤的闪烁光沿光纤传输，送到光接收器，即 CCD 电荷耦合器件。

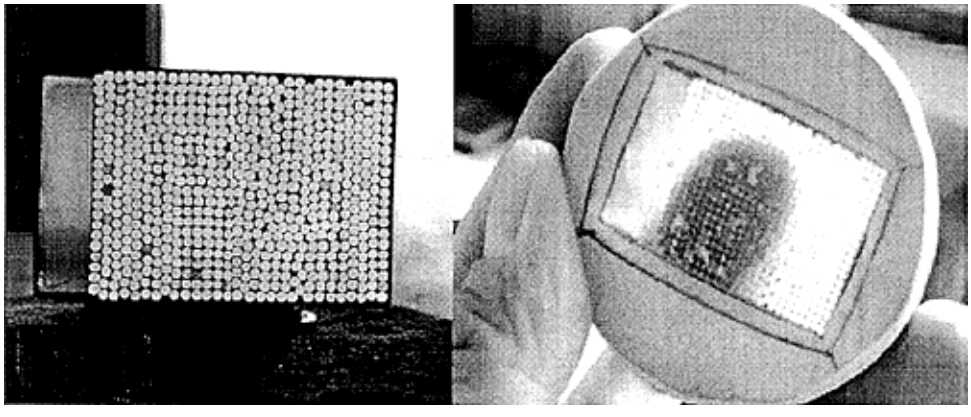


图 2.7　闪烁光纤阵列式探测器实物图

2.2.3 CCD 同步接收与输出

CCD（Charge Coupled Device，电荷耦合器件）是利用集成电路工艺的一种光接受器件，其光敏单元直接将光信号转换成电荷量。将闪烁光纤的光输出末端按照一定的阵列均匀排

放，固定在CCD摄像头的焦平面上，这样就可以将水平阵列的闪烁光纤探测信号转化成一个平面内的视频光点阵列——对光亮度的测量也就转化成对视频电压值的测量。

由于CCD是一种积分型光电转换器件，其输出信号既与光敏单元所受照度有关，也与积分时间（光照时间）有关。这样暴光量过大时，会产生亮度失真或产生大的测量误差，但是光敏单元的照度也不能太低，否则亮度层次少同样会导致信噪比下降。在应用实践中把最大暴光量限制在CCD的饱和暴光量的80%左右比较合适。

在实验系统中我们选择了台湾敏通公司生产的单色高分辨 MTV-1881CCD，其最低照度：0.02Lux，信噪比大于 46dB，饱和输出电压为 1V，工作温度范围-10~50℃。该摄像器件所输出的信号制式是我国黑白电视信号扫描制式：一帧图像的总行数为625，分成两场扫描，主要参数为行频15625HZ（周期64us），场频50HZ（周期20ms），帧频25HZ，为场频的一半（周期40ms）。（黑白全电视信号可以参考图 **2.8**）

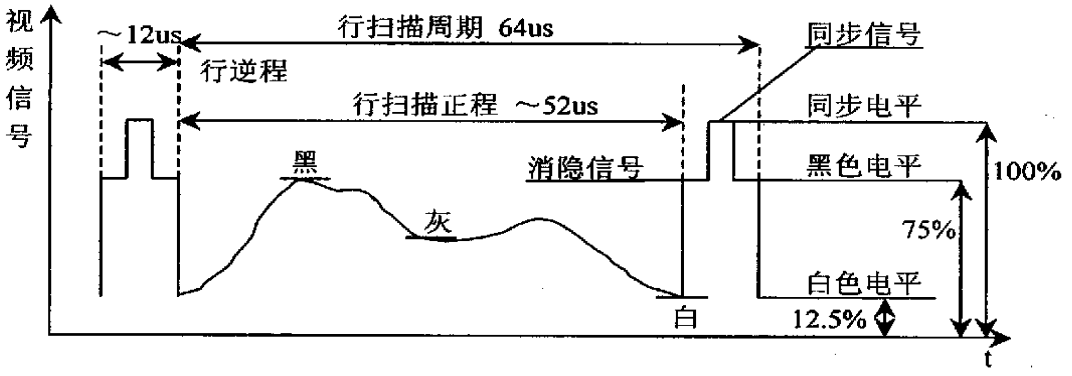


图 **2.8** 黑白全电视视频信号示意图

由于此成像探测器将用于瞬态 X 射线源，因此在设计时必须考虑在射线与闪烁光纤作用后产生的瞬间可见光能被CCD采集到。根据系统这种要求，我们设计了一个基于CPLD的CCD摄像头视频同步信号发生电路，实现了同步信号对CCD采样时刻的控制。

本系统采用的CCD是一种面阵CCD,其内部结构如图 **2.9** 所示：

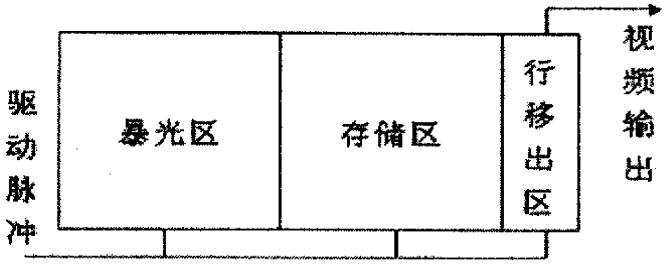


图 **2.9** 面阵CCD内部结构

它的采样、输出由内部驱动电路所产生的时序控制，输出序列（只含控制信号）如图

2.10 所示：

在一个周期 T 内，包括 256 个行同步脉冲，一个场同步脉冲和若干均衡脉冲（无控制作用，此处省去）。场同步脉冲 T1 内，一场视频信号由 CCD 的暴光区移入存储区。在此期间，暴光区做电荷转移操作，采集的光信号无效。行同步脉冲 t1 内，一行视频信号由 CCD 的存储区移入行移出区，此行视频信号在行同步 t1、t2 间输出。在视频信号输入期间，暴光区一直进行光积分，直到下一场同步脉冲 T2 时进行帧移出。

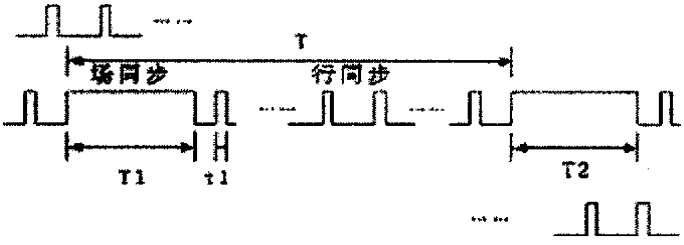


图 2.10 面阵 CCD 控制序列

由上分析可知，在一个周期 T（20ms）内，CCD 在场移出时间 T1 内所采集的信号是无效的。在采集系统中，必须控制该场移出周期所出现的时刻，使其避开有效信号，才能正确、完整的采集到有效的信号。用于采集持续时间特别短（ns 级）的瞬态单次图像信号时，这更具有决定意义。

CCD 摄像头的外部同步信号输入接口，为我们提供了一个捷径。我们只需输入一个如上所示的同步控制信号，即可控制 CCD 内部驱动电路的时序，从而达到控制 CCD 采样时刻的目的，避免了复杂的驱动电路的重新设计。

此 CCD 同步信号发生电路的系统框图如图 2.11 所示。

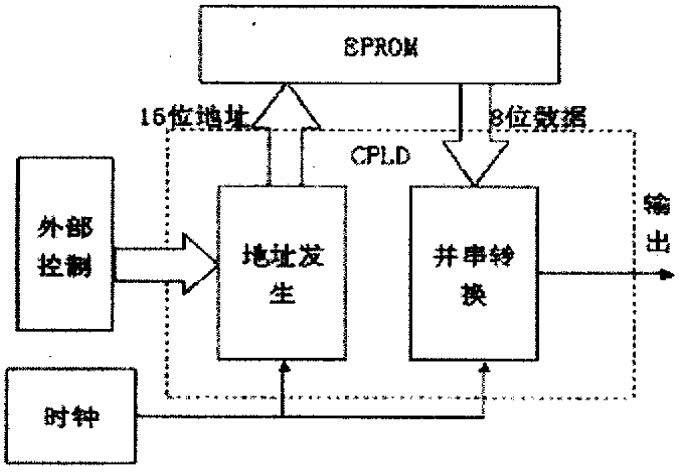


图 2.11 系统构成框图

其中虚线框内部分由 CPLD 实现。

系统主要由以下几个部分构成：

1) 外部时钟。由 10MHz 晶振产生系统时钟，提供给 CPLD 使用。可使同步信号精度达到 0.1us。

2) EPROM。同步信号序列以 10MHz 采样周期量化为二进制位串，存储于 EPROM 中，只需在全局时钟控制下，依次读出 EPROM 中数据，即可产生同步信号。此处同步信号序列周期为 40ms,共需 50KB 存储空间。

3) 地址发生。在全局时钟控制下，产生读取 EPROM 所需的 16 位地址。

4) 串并转换。读出 EPROM 的 8 位数据后，在全局时钟控制下依次串行输出，即可得到同步信号。

5) 外部控制。可自由定义外部控制信号，如 Start、Stop、Reset 等，方便控制同步信号的发生、中止。

本系统有如下优点：

1) 实现简单。

整个系统使用一片 64K EPROM，一片 Lattic 1K 系列的 CPLD，一片 10MHz 晶振即可，容易实现。

2) 通用性。

由于输出信号序列以数字的形式存储于 EPROM 中，改变 EPROM 内容，即可产生需要的输出信号序列。实际上，本系统可作为一个通用的周期信号发生器。

3) 易调试。

CPLD 的引入，结合硬件描述语言的使用，极大的方便了系统的设计、调试。在调试过程中，不需对电路本身进行修改，只需重新烧写 CPLD，变硬件调试为软件调试，大大提高了调试速度。

4) 易扩展。

如需进一步提高信号精度，只需提高全局时钟频率，对地址发生部分做相应扩展即可，通过 CPLD 编程，很容易实现。

另外，还可利用 CPLD，对输出信号做相应处理，进一步扩展系统功能。

5) 控制灵活。

外部输入控制通过 CPLD 编程实现，可定义各种控制信号（Start...）、各种控制方式（电平控制、边缘控制...），可适应各种应用场合。

2.2.4 视频信号的采集

如前所述，在研究中我们对可见光采用了CCD摄像获取数据的技术，将能量信息的获取转化成光强信号数据的采集。我们提出了定时采集整帧视频信号的方法，能够很好地满足系统要求。视频采集模块包括硬件和软件两部分。

2.2.4.1 硬件部分

系统采用的是天敏公司的1000Moons SDK-2000专用视频采集卡，这是一款专门针对系统开发商及电脑DIY发烧友的高品质PCI视频卡。它具有高品质的视频采集性能，具备高速PCI总线，兼容即插即用（PNP），支持一机多卡。它给开发人员提供了功能全面的二次开发包(SDK)，开发人员可以选择VisualBasic、VisualC++、Delphi等多种编程语言通过SDK进行开发，SDK中包含DLL动态库（VC使用），OCX控件(VB,Delphi使用)及详细说明。可以通过SDK控制图像的输入端口，图像亮度，对比度，色度，灰度等输入信号，动态截取图像，以AVI格式进行录像侦测图像是否有移动目标等等。

此采集卡基于PCI总线，兼容Windows即插即用，安装简易；显示画面流畅，采集速度可达30帧/秒；显示分辨率可达640x480，24位真彩；影像窗口大小随意调整，可全屏显示；动态捕捉影像以静态图像方式存盘，提供BMP,JPG,PCX,GIF,TIF,TGA等多种存盘格式；提供动态AVI影像捕获；兼容Windows VFW软件架构和WDM模式，提供功能全面的二次开发包，可广泛应用于监控及远程通信等方面的系统开发。

此款视频采集卡对系统需求不高：奔腾100MHz或以上处理器；16Mb或以上的系统内存；支持DirectX的VGA显示卡；Windows95/Windows98或以上的视窗操作系统。接口说明如下图所示：

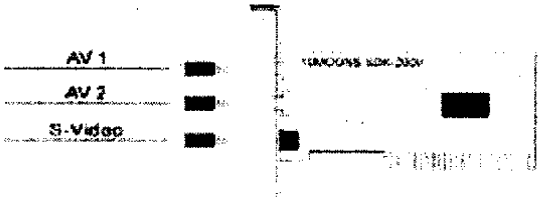


图 2.12 10Moons SDK2000视频采集卡接口示意图

2.2.4.2 软件开发

图像采集

由于此视频采集卡支持二次开发，因此我们可以根据探测器系统要求自行开发定时采集

程序。根据系统要求，需要对软件要求提供外部并口控制采集过程功能，并且提供预览，采集单帧图像，动态图像，分解动态图像，设置采集区域，采集速率，采集制式等功能。功能的实现主要通过调用视频采集卡提供的设备驱动程序提供的调用接口和控制计算机并口提供的数据端口信号输出控制外部设备。整个采集过程包括：1）计算机发出并口控制信号，控制器启动 CCD 同步信号，并使计算机启动视频采集卡。2）计算机发出并口控制信号，打开 Xray 设备。3）计算机采集图像保存。4）关闭所有设备，结束。

为了方便系统扩展，做好二次开发，以及测试的方便，采用了 MCV(Model-Control-View) 结构体系：

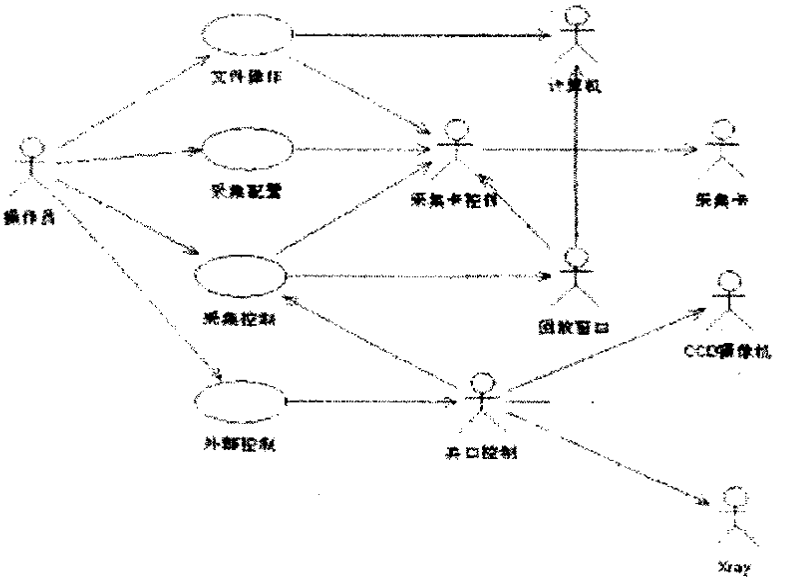


图 2.13 视频采集软件 MCV 结构体系

采集程序分为三个独立运行的模块：采集控制，设备控制，图像分解。执行程序将在 Windows 2000/9x 上运行。硬件方面，本系统要求使用采集卡，CCD 摄像头，光纤探测器等外设，通过配置一般的台式机均具备运行本系统的最低硬件要求。开发语言我们选择的是 VC++和 Delphi。开发时需要安装视频采集卡自带的二次开发软件包以及一些控件。

由于使用了开发商提供的控件，以及自主开发的控件，我们将对控件做一次封装实时显示采集图像，并且提供采集信号控制功能和分解回放片断，所以我们将软件分为“采集控制”窗口，“采集窗口”，“回放窗口”，“并口控制组件”四个模块。程序启动后将显示采集控制窗口和采集窗口；回放窗口由采集控制窗口的菜单提供显示功能；并口控制组件控制定义并口引脚的时序，从而达到控制外部设备的目的。下面我们对这四个模块的功能进行描述。

1）采集控制

主要功能：采集图像的文件处理；采集控制；设置采集卡；控制外部采集仪器；帮助功

能。采集图像的文件处理主要功能包括当前帧图像的保存；剪贴板图像保存；设置视频保存文件名；保存当前帧到剪贴板。采集控制的功能包括预览，即将采集卡的采集图像信息通过CPU 处理后再显示出来；冻结，即采集窗口的画面帧不在变化；覆盖，即将采集卡的采集图像信息直接送至显卡的显存中显示；开始采集，即采集窗口在没有外部触发控制采集时，本选项可以使程序开始采集视频；停止采集，即采集窗口在没有外部触发控制采集时,本选项可以使程序停止采集视频；手工采集，即采集窗口在没有外部触发控制采集时,本选项可以使程序手工控制停止采集视频；回放片段，即回放刚刚的采集片段。设置采集卡主要功能包括：采集帧数/次，即确定后通知采集窗口设置采集卡的捕捉时间；采集速率，即确定后通知采集窗口设置采集卡每秒中采集多少帧，由于采集卡性能的限制，最大只为30帧；视频格式，即通知采集窗口显示设置对话框，设置采集卡的图形格式:24/15 位RGB,YUY2,YUV9/12 以及捕捉大小选择；视频来源，即通知采集窗口显示设置对话框，设置采集卡的视频输入 AV1/2,S 端子;视频制式:P/N/S 制以及颜色的设置；视频压缩，即通知采集窗口显示设置对话框，设置采集文件的压缩格式,为保证图像不损失,使用不压缩；窗口恢复为采集尺寸，即通知采集窗口大小恢复采集大小。外部控制主要功能是通过调用二次开发组件，启停外部控制，显示外部控制设置窗口。

2）采集窗口

封装了提供商控件，用来显示采集状态和显示采集图像的窗口，并且提供采集功能函数的调用。

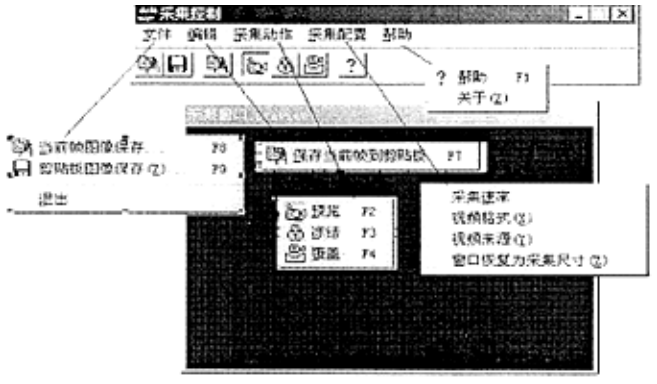
3）回放窗口

用来浏览采集的片段的窗口并可从中选取所需的单帧图像保存。1)显示前一帧图像 2)显示下一帧图像 3）保存当前帧图像。

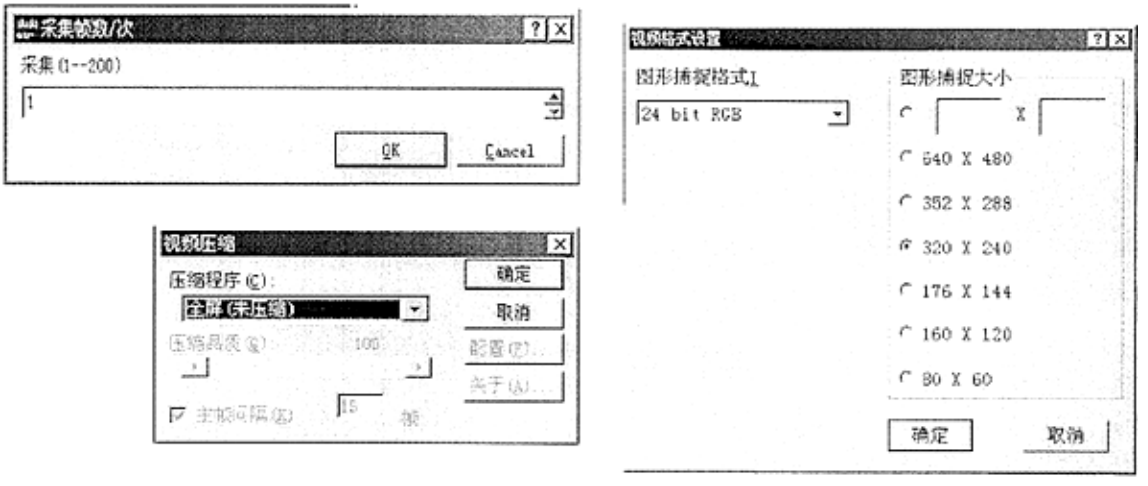
4）并口控制

一般计算机都配备有并行打印机端口，它是一个25芯的母接头，通常工作于8位数据输出方式用于驱动打印机等外设，同时可对这些设备的状态进行监测，确定工作状态。本程序中使用了 LPT1 口的8位数据位控制采集过程，在NT环境下，内核安全模式禁止直接对端口进行操作。但在用 VC++编程时， 可以利用 NT 的 API 函数 CreateFile 和 WriteFile 控制操作。应该注意的，必须将 DB25 输入插座的 11 脚和 12 脚接地，否则，操作不能顺利进行。通过对并口数据位的控制，可以达到控制外部设备的目的，从而协调整个系统的工作时序。

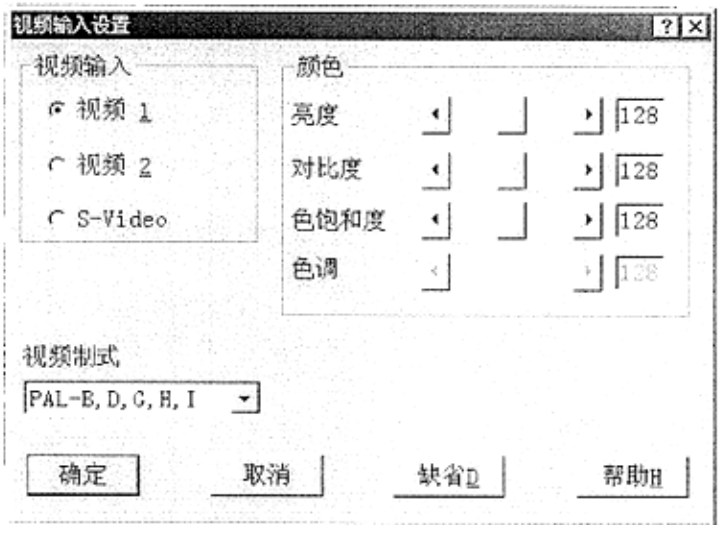
此采集软件的部分操作界面可以参考图 2.14。


采集控制
文件 编辑 采集动作 采集配置 帮助
? 帮助 F1
关于(Z)
当前帧图像保存... F9
剪贴板图像保存(Z) F9
退出
保存当前帧到剪贴板 F7
拍照 F2
连续 F3
预览 F4
采集速率
视频格式(Z)
视频来源(I)
窗口恢复为采集尺寸(Z)

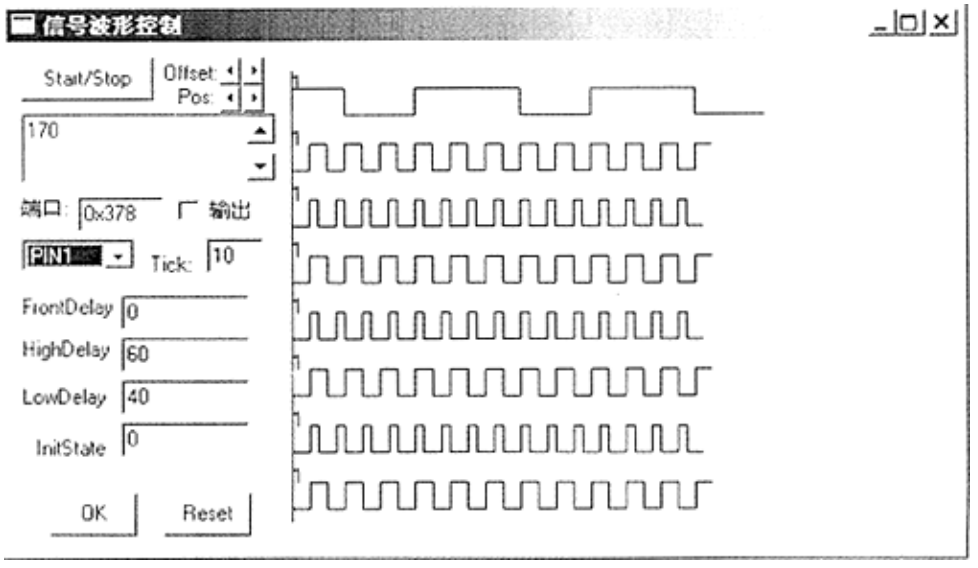
(a)


采集帧数/次
采集(1--200)
1
OK
Cancel
视频压缩
压缩程序(C):
全屏(无压缩)
压缩品质(Q)
100
关键帧间隔(K)
15
帧
确定
取消
配置(C)...
关于(A)...
视频格式设置
图形捕捉格式I
24 bit RGB
图形捕捉大小
X
640 X 480
352 X 288
320 X 240
176 X 144
160 X 120
80 X 60
确定
取消

(b)


视频输入设置
视频输入
视频 1
视频 2
S-Video
颜色
亮度
对比度
色饱和度
色调
128
128
128
128
视频制式
PAL-B, D, G, H, I
确定
取消
缺省D
帮助H

(c)



(d)

图 2.14 视频采集软件部分操作界面

(a)采集控制和采集窗口主界面(b)采集速率，视频压缩和视频格式设置(c)视频输入设置(d)并口 8 个数据位的时序波形控制

图像处理

由于系统内部和外部存在各种类型的噪声，因此在进行图像采集后必须要进行图像处理，以达到降低噪声，增强系统信噪比，提高图像质量的目的。降低噪声的图像处理方法有许多种，包括多幅叠加平均法，卷积法，频域滤波法等。我们第一步选择了其中一种较为普通的方法——多幅叠加平均法[15]。

一般来说，射线数字成像中的随机噪声服从高斯分布，因此，提高信噪比最为有效的措施之一是多幅图像的叠加平均。下面给出了这一方法有效性的相关证明。

设多幅静止图像 $D_i(x,y)$ 被噪声 $N_i(x,y)$ 污染，则有：

$$D_i(x,y)=S(x,y)+N_i(x,y)$$

噪声的特点（性质）：噪声的期望为 0，即：

$$E\{N_i(x,y)\}=0 \tag{2.5}$$

$$E\{N_i(x,y)+N_j(x,y)\}=E\{N_i(x,y)\}+E\{N_j(x,y)\} \tag{2.6}$$

$$E\{N_i(x,y)N_j(x,y)\}=E\{N_i(x,y)\}E\{N_j(x,y)\} \tag{2.7}$$

点(x, y)的功率信噪比：

$$P(x,y)=\frac{S^2(x,y)}{E\{N^2(x,y)\}}$$

对M幅图像作平均：

$$\bar{D}(x,y)=\frac{1}{M}\sum_{i=1}^{M}[S(x,y)+N_i(x,y)]$$

$$\bar{D}(x,y)=S(x,y)+\frac{1}{M}\sum_{i=1}^{M}N_i(x,y)$$

平方信噪比：

$$\bar{P}(x,y)=\frac{S^2(x,y)}{E\{[\frac{1}{M}\sum_{i=1}^{M}N_i(x,y)]^2\}}$$

$$\bar{P}(x,y)=\frac{M^2S^2(x,y)}{E\{\sum_{i=1}^{M}\sum_{j=1}^{M}N_i(x,y)N_j(x,y)\}}$$

$$\bar{P}(x,y)=\frac{M^2S^2(x,y)}{E\{\sum_{i=1}^{M}N_i^2(x,y)\}+E\{\sum_{i=1}^{M}\sum_{j=1}^{M}N_i(x,y)N_j(x,y)\}}$$

由式(2.5)，(2.7)得：

$$\bar{P}(x,y)=\frac{M^2S^2(x,y)}{\sum_{i=1}^{M}E\{N_i^2(x,y)\}}$$

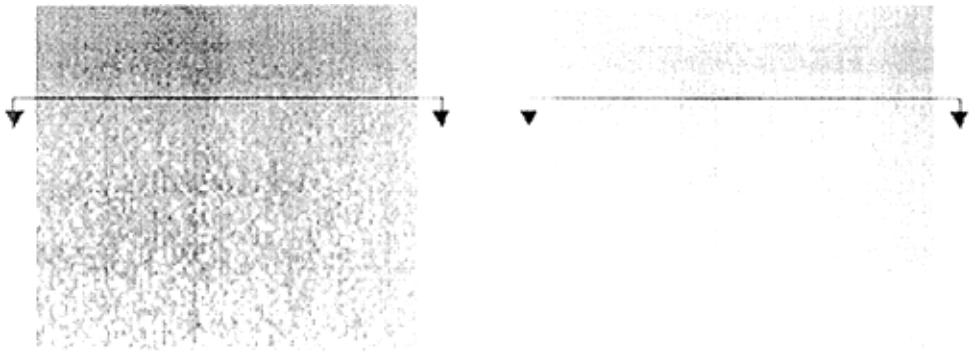
$$\bar{P}(x,y)=\frac{M^2S^2(x,y)}{ME\{N^2(x,y)\}}$$

$$\bar{P}(x,y)=\frac{MS^2(x,y)}{E\{N^2(x,y)\}}$$

$$\bar{P}(x,y)=MP(x,y) \qquad (2.8)$$

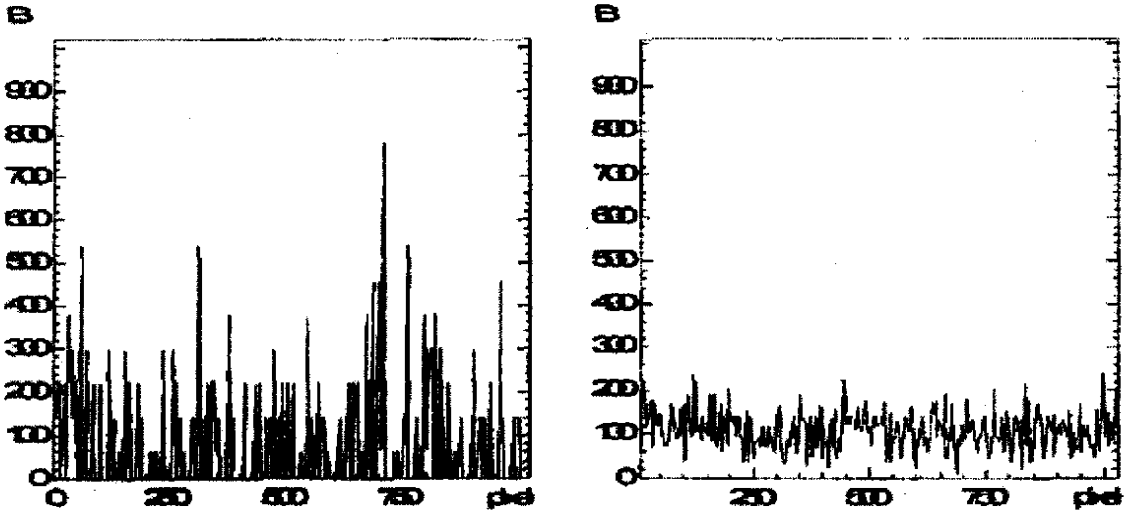
式(2.8)证明了：对M幅图像进行平均，使图像中每一点平方信噪比提高了M倍，信噪比提高了$\sqrt{M}$倍。

图2.15是在管电压60kV下拍摄得均匀射线成像图像，图2.16为对应图像某一行得灰度曲线。从图中叠加前后得图像对比可以看出，原始图像中存在得白噪声得到了很好得抑止，图像质量明显改善。



(a)原始图像　　(b)16幅平均后的

图 2.15 叠加平均前后图像比较



（a）原始图像　　（b）　16幅叠加平均图像

图 2.16 图像某一行灰度曲线比较

多幅图像叠加平均前原始图像(1024x1024)第 500 行标准差为 935.72，16 幅图像叠加平均后为 218.02，降低了 76.7%。

在我们这个系统中，图像叠加平均，即在指定的时间内，选取组成一个 avi 动态图像文件的多幅静态图像进行象素灰度值的相加平均来去处白噪声，以提高信噪比。为此，我们专门编写了一个图像叠加处理软件，选择的语言环境是 Delphi，程序运行环境为 Windows2000。软件操作界面如图 2.17 所示。

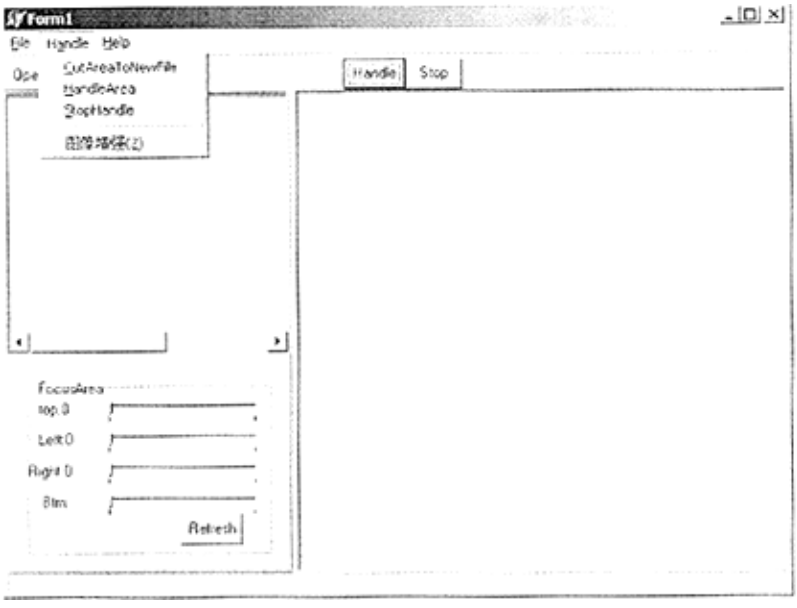
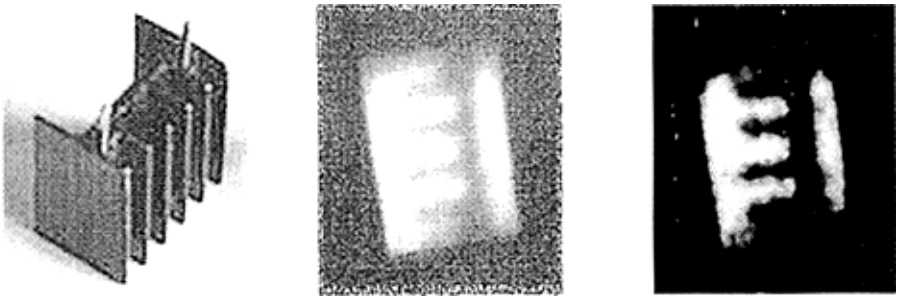


图 2.17 图像叠加平均处理软件操作界面

在对图像做完叠加平均处理之后，我们可以再利用现成的专业图像处理软件 PhotoShop 来对图像进行容差，图像增强等处理，以进一步降低图像噪声，提高信噪比。图 2.18 为被测物体为散热片时采集后的图像与处理后的图像对比示例。



实物图　　采集后　　处理后

图 2.18 图像采集后与处理后的举例比较

2.2.5 实验工作

此闪烁光纤探测器系统将用于较大范围能量下的成像，但由于条件所限，因此实验工作暂时只能在低能下进行。本系统实验采用的 X 射线源是牙科医用的 X 光机，其管压 35kV，电流 5mA，暴光时间 2s。其它主要实验设备包括台湾敏通公司生产的单色高分辨 MTV-1881CCD；自制变档放大器，最高放大倍数达到 100 倍，实验时选择放大 20 倍；视频采集卡；计算机以及一个 20x35 的闪烁光纤阵列探测器。视频采集程序设置：采集速率:30

帧/秒；PAL 制；RGB 格式；亮度 :0 ；对比度 :251 (0-255) ；采集面积(单位:象素):640*480；无压缩算法；采集口:1 号口；文件格式:avi 文件；其他为默认参数。

探测器系统结构示意图如图 2.19 所示。为了避免 CCD 不被 X 射线因直射而导致 CCD 镜头受损，因此在实验时，我们在 CCD 与射线源之间加入一个反射平面镜，X 射线通过反射到达 CCD，从而减少射线强度，起到保护 CCD 的效果。

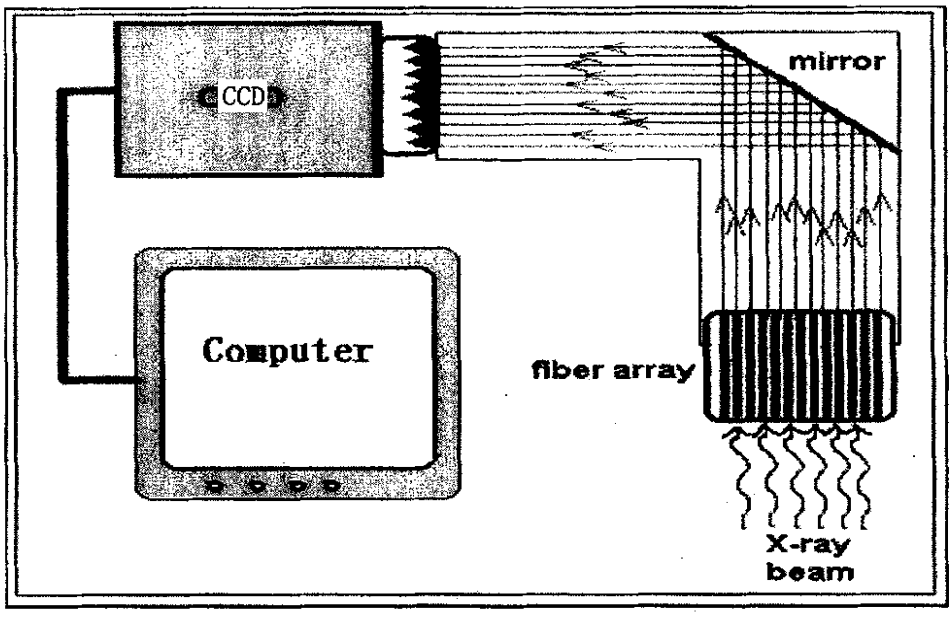


图 2.19 闪烁光纤成像探测器系统结构示意图

通过对闪烁光纤成像探测器系统设计的测试结果来看，在这种能量范围下可以达到了预期的效果，因此证明在较低能 X 射线下，采用塑料闪烁光纤作为成像探测器是完全可行的。较其他成像探测器而言，由于单根闪烁光纤的直径可以达到很小，即图像的每个象素单元尺寸很小，这样使得图像的分辨率可以达到较高，再加上在低能下，阵列中光纤之间的散射粒子的串扰程度不大，使得系统最终的空间分辨率在较大程度上取决于闪烁光纤的直径大小。因此在低能 X 射线下，闪烁光纤成像探测器具有空间分辨率较好的优点。而此探测器在较高能下应用的可行性研究，我们将在下面章节的模拟计算分析中解释。

第三章 射线成像探测器的基本特性概念

对于一种射线成像探测器，若要定量的来描述其某些特性，我们可以利用相对应的某些参数来表达，从而可以达到定性的分析探测器各个方面的优劣性，这些参数我们称之为探测器“特性参数”。例如，前面章节提到的探测器吸收效率，空间分辨率，信噪比等都是属于此类参数。在这一章节中，我们将着重对几个常用的探测器特性参数概念作一个详尽的分析解释。

3.1 探测效率(DE)

作为选择探测器最重要的衡量标准，探测效率（Detector Efficiency）反映了系统灵敏度。它与系统许多因素有关，包括闪烁体，光电设备，被测物体厚度等。这其中，以闪烁体因素最为重要，它的探测效率决定着成像探测器的总体光产额。闪烁体的射线探测效率主要与自身材料厚度，射线源能量以及被测物体厚度有关。对于用于不同能量范围下，我们必须选择在此能量下灵敏度最高的闪烁体，从而有利于最终图像的质量。

闪烁体探测效率(DE)的定义为[18]：

$$DE = \frac{\text{输出闪烁光能量}}{\text{输入光子能量}} \qquad (3.1)$$

总体来说，在闪烁体中，无机闪烁体的探测效率比有机闪烁体高。这是由于无机闪烁体密度大，含有高原子序数的元素，因而其发光效率高，能量正比关系好。例如，在常用的无机闪烁体中，NaI(Tl)晶体密度 $\rho = 3.67 g/cm^3$，含有高原子序数的碘(Z=53)，是X射线良好的吸收体。它的光产额高，每keV能量平均产生40个可见光光子，输出的闪烁光信号强，能量分辨率是所有实用的闪烁体中最好的一种。NaI（Tl）的发光衰减时间室温下是 $0.23\mu s$，光谱主峰位为 $4150\,\mathring{A}$ 左右，半高宽约为 $850\,\mathring{A}$，它的吸收谱峰值在 $2930\,\mathring{A}$ 和 $2340\,\mathring{A}$，因此对所发射的光是透明的。CsI（Tl）晶体发光效率比NaI（Tl）低，每keV能量平均产生16个可见光光子，它的光谱分布很广，从蓝光到红光都有，主峰位在 $5400\,\mathring{A}$。它的密度（$4.51 g/cm^3$）和平均原子序数都比NaI（Tl）大，对X射线吸收系数更大，探

高。

有机闪烁体由于主要由C和H元素构成，因此平均原子序数不大，因而就探测效率而言不如无机闪烁体。目前，许多有机闪烁体里参有少量重金属或某些高原子序数的闪烁体，其目的之一就是为了提高有机闪烁体的探测效率。在有机闪烁体中，最为常用的是闪烁光纤。以本系统采用的闪烁光纤 BCF-20 为例，其发光衰减时间室温下是 2.7ns，光谱主峰位为492nm左右，它每keV能量产生8个可见光光子。闪烁体光纤的探测效率的定义为[18]：

$$DE=\frac{\text{输出闪烁光能量}}{\text{输入光子能量}}=Ta\times Tc\times Tt \qquad (3.2)$$

式中Ta是闪烁光纤的光子能量吸收效率，它反映有多少注入的光子能量转变为激励起二次电子跃迁的吸收损耗，即光纤里有多少能量沉积，它的值应为：

$$Ta=\frac{\text{吸收的光子能量}}{\text{注入的光子能量}} \qquad (3.3)$$

Tc 是闪烁光纤的光转换效率，又称量子效率或发光效率，它表明有多少吸收的光子能量转变为闪烁光，它的大小为：

$$Tc=\frac{\text{产生的闪烁光能量}}{\text{吸收的光子能量}} \qquad (3.4)$$

Tt 是闪烁光纤的传输效率，它表明产生的闪烁光有多少可被光纤捕获而在光纤内传输，其定义为：

$$Tt=\frac{\text{输出的闪烁光能量}}{\text{产生的闪烁光能量}} \qquad (3.5)$$

可以从图 3.1 中看出 Ta，Tc，Tt 三者关系。

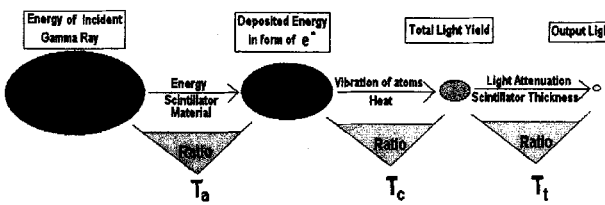


图 3.1 闪烁光纤探测效率参数 Ta，Tc，Tt 三者关系

在射线成像系统中，入射的光子，X或γ射线等，都是电磁辐射。它们与物质的相互作用主要是与电子的离散碰撞。在光子能量为10keV---20MeV的范围内，占优势的相互作用是

光电吸收、电子对的产生和康普敦散射。因此，光子能量吸收应包含注入的光子能量转变为二次电子的动能。这时，光子能量吸收效率可近似表示为[27]

$$\begin{aligned} T_a &= \frac{I(t)}{I_0} \\ &= 1 - \exp[-\sum \int_0^{E_{max}} \mu_i(E)tdE] \end{aligned} \tag{3.6}$$

式中，I。是注入的光子强度，t是相互作用物质的厚度，μi(E)是I材料的能量吸收系数。

由式(3.6)可知，较厚的探测器和较高的μi(E)值，将使探测器具有较高的吸收效率Ta，又因相互作用率N与吸收效率Ta成正比，以及泊松统计支配着辐射探测，所以探测器的信噪比(SNR)与相互作用率N的平方根成正比。即

$$N \propto T_a, \quad SNR \propto \frac{N}{\sqrt{N}} = \sqrt{N} \tag{3.7}$$

可见，对于给定的曝光周期，具有较高吸收效率Ta的探测器，将会具有较高的信噪比。

由于被激励起的闪烁光是各向同性的，同时只有位于临界角θc以外的那些闪烁光才能在光纤内被反射并传播，所以可传输的闪烁光总量大约为 1−n1/n2，此光的一半传输至接有光电倍增PMT的右端面，传输的平均距离为L*1/2；另一半光则传输至左端面后被反射再传输至右端面。此时传输的平均距离为L*3/2，显然，此时的传输效率应为[28]

$$\begin{aligned} T_t &= T_{t_\perp} + T_{t_{//}} \\ &= \frac{1}{2}(1 - \frac{n_1}{n_2})\exp(-\frac{L}{2a}) + \frac{1}{2}(1 - \frac{n_1}{n_2}) \\ &\quad \times \exp(-\frac{3L}{2a})(1 - \frac{t_\perp + t_{//}}{2}) \end{aligned} \tag{3.8}$$

式中，L是闪烁光纤的长度，a是反映吸收损耗和反射损耗的衰减系数，t⊥和t//是闪烁光纤左端处的垂直分量和水平分量的透射率。它们的大小分别为

$$\left.\begin{aligned} t_\perp &= \frac{4n_1\cos\theta(n_0^2 - n_1^2\sin^2\theta)^{1/2}}{[n_1\cos\theta + (n_0^2 - n_1^2\sin^2\theta)^{1/2}]^2} \\ t_{//} &= \frac{4n_0n_1\cos\theta(n_0^2 - n_1^2\sin^2\theta)^{1/2}}{[n_0^2\cos\theta + n_1(n_0^2 - n_1^2\sin^2\theta)^{1/2}]^2} \end{aligned}\right\} \tag{3.9}$$

其中，no是闪烁光纤左端面外部介质的折射率，θ是相对于光纤轴向的平均角度。

3.2 信噪比（SNR）

信噪比（Signal-to-Noise Ratio）的定义为系统有效信号与系统噪声的比值。在任何一种射线成像系统中，探测器信噪比是决定图像质量，衡量噪声水平的重要指标。换句话说，在一个射线成像系统中，一切工作的目的都是为了得到一个信噪比高的图像，由于在系统内部

和外部必然存在各种随机或固有的，低频的或高频的噪声，因此我们必须采取各种抑止噪声的措施来提高信噪比。但由于在射线成像系统中，光子传输具有随机效应，本身存在统计起伏，这些统计起伏产生的噪声是系统必然存在的，这种噪声我们称之为量子噪声。量子噪声是X射线量子依照泊松(Poisson)分布的统计学法则随机产生的空间波动。量子噪声的大小与探测器检测到的X射线成反比，即与入射到探测器的射线量成反比。入射的（检测到的）X射线剂量越大，量子噪声越小，其大小通常以射线剂量的均值平方根 RMS（Root Mean Square）表示。在低剂量条件下，RMS 值对射线辐射剂量相应的变化近似直线递减，表明噪声主要由于X射线能量的波动（量子噪声）引起；在高剂量条件下，RMS 大致接近一恒定值，几乎不依赖于辐射剂量，表明非量子噪声成为决定性因素。根据以上叙述，若入射的X射线剂量在允许剂量下限之上且恒定时，则成像系统噪声大小由探测器的吸收特性决定。因此提高探测器的动态范围，即信噪比，可以在抑止量子噪声的同时提高图像的质量。根据上述情况，假设有N个入射光子被探测器检测到，则此时的信噪比 $SNR = N/\sqrt{N} = \sqrt{N}$ 。

除了量子噪声以外，在X射线数字成像中，由于X射线散射难以避免地对CCD器件的激励作用，会在获取的图像上产生随机分布的脉冲噪声。虽然CCD对X射线的响应性能比对可见光的差，但仍能吸收X射线，形成载流子而污染信号。由于散射到CCD上的单个射线光子的能量与可见光子能量相比要大得多，所以当一个X射线光子射到CCD某个像元上时，该像元比其邻近的像元会收集到更多数量的电荷。尤其在微光条件下，这些由X射线激发的电荷反映到图像上就形成了幅度较大的脉冲噪声，这种脉冲噪声对图像的影响非常严重。因为CCD的感光面积很小，而散射X射线是随机的，到达某个CCD像元的概率也是随机的，因此可将散射射线对CCD采集到的图像的影响看作是一种空间上的随机分布的正向脉冲噪声，而且与邻近的未受污染区域的像元灰度无关。由于散射射线产生的噪声具有不同于高斯噪声的性质，其滤波算法也应不同于高斯噪声的算法。为了降低X射线对探测器的直接影响，应当加强对探测器的射线屏蔽。

探测器信噪比包括输入信噪比($SNR_i = S_i / N_i$)和输出信噪比($SNR_O = S_O / N_O$)。在成像系统中，一般情况下，由于在经过系统传输后，信号有不同程度的损失，因此输出信噪比要小于输入信噪比，若两者比值越接近1，则表明探测器系统越理想[11]。

3.3 探测量子效率(DQE)

探测量子效率(Detective Quantum Efficiency)是反映探测器信息传输效率的一个参数[19]。其定义为输出与输入信噪比平方的比值。

$$DQE = SNR_o^2 \Big/ SNR_i^2 \tag{3.10}$$

从式(3.10)可以看出，DQE实际上是描述了探测设备保存从放射区域到得到的数字成像数据的信噪比能力，是指在图像探测器入口的输入信号的噪声功率谱与反馈回输入端的输出信号的噪声功率谱的比值。换句话说，是系统信噪比平方的传递函数。它是衡量探测器系统性能的重要指标。但DQE只能反映系统信噪比的在传输中的损耗程度，并不能直接反映图像质量，图像质量只有通过信噪比的绝对大小来反映。

当只讨论探测器系统本身，即不考虑被测物体存在时，此时的DQE值是处于零频域下；若要考虑存在被测物体的情况时，我们可以通过频域下的DQE函数DQE(f)来衡量成像系统的性能。零频域的DQE值直接影响DQE(f)函数。

在前一节我们曾经提到平均相互作用率的概念。具体的说，探测器与粒子的平均相互作用率描述的是在一定剂量下探测器与粒子发生相互作用（光电效应，康普敦散射，电子对效应）的平均次数，我们把这个平均相互作用率用参数——量子效率 QE（Quantum Efficiency）来表示。

从以上DQE和QE两个概念可以看出，QE是DQE的特殊情况，即当探测器系统只考虑输入端的泊松统计噪声和由于相互作用率<1 而产生的二项式统计噪声，而不考虑其它噪声因素时，DQE=QE。我们称此时的探测器为“准理想探测器”[11]。而在实际探测器系统中，DQE<QE<1。

由于“准理想探测器”只考虑输入端泊松统计噪声和由于QE<1 而导致的二项式统计噪声，而这两种噪声是独立的，无关联的，因此准理想探测器输出的总噪声为：

$$\sigma_o^2 = (\sigma_i)_o^2 + \sigma_{QE}^2 \tag{3.11}$$

其中$(\sigma_i)_o^2$为从探测器输出角度看输入噪声$(\sigma_i)^2$，σ_{QE}^2为由于QE<1 而导致的二项式统计噪声。又因为：

$$(\sigma_i)_o^2 = QE^2\sigma_i^2 = QE^2\overline{S_i} \qquad \sigma_{QE}^2 = QE(1-QE)\overline{S_i}$$

S_i为平均输入信号，具有泊松波动。因此：

$$\sigma_o^2 = QE^2\overline{S_i} + QE(1-QE)\overline{S_i} = QE\overline{S_i} = \overline{S_o} \tag{3.12}$$

若 QE=1，则$\sigma_i^2 = \sigma_o^2 = \overline{S_i} = \overline{S_o}$。

实际上，
$$\left(\frac{\overline{S_o}}{\sigma_o}\right)^2 = \frac{QE^2\overline{S_i}^2}{QE\overline{S_i}} = QE\overline{S_i} = QE\frac{\overline{S_i}^2}{\overline{S_i}} = QE\left(\frac{\overline{S_i}}{\sigma_i}\right)^2 \tag{3.13}$$

所以

$$QE = \frac{\left(\frac{\overline{S_o}}{\sigma_o}\right)^2}{\left(\frac{\overline{S_i}}{\sigma_i}\right)^2} = \frac{\overline{S_o}}{\overline{S_i}} \tag{3.14}$$

这些有关准理想探测器 QE 的算法实际上都包含了 DQE 的概念，因此对 DQE 而言也非常重要。尤其式(3.14)中，噪声的计算是以输入输出信号的波动为依据的。

与准理想探测器比较，在实际探测器系统中，还有一个附加的噪声源σ_n^2。此时的输出噪声变为：

$$\sigma_o^2 = (\sigma_i)_o^2 + \sigma_{QE}^2 + \sigma_n^2 \tag{3.15}$$

在实际探测器中的 QE 与准理想探测器是一样的。为了获得S_o，我们必须把附加的噪声源从系统输出中减去，因为这些噪声源是恒定的，不受系统影响，平均值为零。因此，对于实际探测器来说，有

$$\overline{S_o} = QE\overline{S_i}$$

根据式（3.12）得到：

$$\sigma_o^2 = \overline{S_o} + \sigma_n^2 \tag{3.16}$$

这里的$\overline{S_o}$是以量子数为表达。因此，实际探测器系统的SNR_{out}要比准理想探测器系统小，但两者的 QE 和SNR_{in}是相同的。

根据式(3.10)，我么可以得到实际探测器 DQE 的具体表达式：

$$DQE=\frac{\left(\frac{\overline{S_o}}{\sigma_n}\right)^2}{\left(\frac{\overline{S_i}}{\sigma_i}\right)^2} \tag{3.17}$$

在实际中，我们常常通过 DQE 把实际探测器的性能转换为准探测器来表达，这样可以对不同成像探测器的性能进行有效的比较。因为 QE 与准理想探测器的 DQE 在概念上是完全一致的。

利用公式(3.16)，DQE 又可以转化为：

$$DQE=\frac{\frac{\overline{S_o}^2}{\overline{S_o}+\sigma_n^2}}{\overline{S_i}}=\frac{QE}{1+\frac{\sigma_n^2}{QE\overline{S_i}}} \tag{3.18}$$

与 QE 不同，DQE 不是一个简单的探测器特性常数，它不仅取决于 QE，还取决于平均输入信号 S_i（输入泊松分布噪声）以及成像探测器附加噪声源 σ_n^2。

当考虑被测物体对成像探测器系统性能影响时，这时我们应该考察频域下的 DQE 值，即 DQE(f)函数。也就是说我们必须把 DQE 从时域转化到频域，这时必须引入功率谱密度的概念。功率谱为在以中心频率为 f 的一个归一化的带宽内，信号（或噪声）的平均功率分布。因此我们目标就是计算或测量 DQE 在空间频率为 f_x, f_y 的值，f_x, f_y 为沿空间坐标 x,y 方向上的功率谱。

根据式(3.16)和(3.18)，我们可以把式(3.17)改写成

$$DQE(f)=\frac{\frac{\overline{S_o}^2}{\overline{S_o}+\sigma_n^2}}{\frac{\overline{S_i^2}}{\overline{S_i}}}=\frac{\frac{QE^2\overline{S_i^2}}{QE\overline{S_i}+\sigma_n^2}}{\frac{\overline{S_i^2}}{\overline{S_i}}} \tag{3.19}$$

在公式中，频率 f 包含沿 x,y 方向两个分量，但为了简化计算，我们仅以 X 方向的分量 f_x 代表整个 f，然后对带宽进行归一化，并以 f 中心频率。则公式(3.19)变为：

$$DQE(f)=\frac{\frac{QE^2W_i(f)MTF(f)^2}{QEW_{ni}(f)MTF(f)^2+W_n(f)}}{\frac{W_i(f)}{W_{ni}(f)}} \tag{3.20}$$

这里的 $W_i(f)$和 $W_{ni}(f)$分别为为探测器输入端信号和噪声的功率谱，$W_n(f)$为附加噪声的功率谱。公式中的 MTF^2 为系统调制传递函数，我们将在后面一节中具体解释其含义。它可以通过系统对 $W_i(f)$和 $W_{ni}(f)$进行过滤。则：

$$DQE(f)=\frac{QE}{1+\dfrac{W_n(f)}{QEW_{ni}(f)MTF(f)^2}} \tag{3.21}$$

成像探测器的 DQE 是在时域或频域下以量子效率的观点来处理的。因此采用 DQE 的量子效率观点是合理解释 DQE 公式或测量当中的一些有关参数含义的实质。对频域下 DQE 的这种处理可以推出时域下 DQE 的结论。总而言之，探测量子效应 DQE 被认为是衡量成像探测器灵敏度和噪声性能最有用的理论。其实质就是入射到成像系统中产生图像的那部分有效入射粒子。作为反映被测物体细节的频域下 DQE 公式为：

$$DQE(f)=\frac{SNR_{out}^2}{SNR_{in}^2}=\frac{\overline{S}^2MTF^2(f)}{\Phi NPS(f)} \tag{3.22}$$

这里的 SNR_{in} 和 SNR_{out} 分别为探测器输入端和探测器输出端图像的量子信噪比。S 为平均图像信号，Φ 总入射粒子数，MTF 为成像系统的调制传递函数，NPS 为图像的噪声功率谱。

MTF 描述的是频域下信号的衰减情况，它是对线扩展函数(LSF)进行傅立叶变换得到的。

$$MTF(f)=\left|FFT[LSF]\right|=\left|\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}LSF(x)e^{i2\pi fx}dx\right| \tag{3.23}$$

3.4 调制传递函数(MTF)

调制传递函数 MTF(Modulate Transfer Function)是不同图像空间频率下，成像系统的响应函数，它用来衡量比较图像系统的空间分辨率。它也是描述数字成像系统的最好工具，这在以下我们将做具体解释。

MTF 是点扩展函数(PSF)傅立叶变换的标准定义。它是一个二维量，对应于二维图像，但在实际测量时，我们仅仅单独考虑一个方向，即仅对图像做一维处理。

测量MTF方法一般有四种：针孔照相法、台阶照相法、狭缝照相法和分辨率板照相法[15]。

在这里我们采用了台阶法测量光纤阵列系统的MTF值。点扩展函数(PSF,用h表示)、线扩展函数(LSF,用L表示)和调制传递函数(MTF,用I表示)是三个描述成像系统空间分辨率的最科学的参数[31]。PSF 和MTF可独自表述成像系统的质量,它们之间可相互转换，两者是傅立叶变换对。即，

$$\mathrm{PSF} \overset{FT}{\underset{FT^{-1}}{\Leftrightarrow}} \mathrm{MTF} \tag{3.24}$$

LSF 也是描述成像系统质量的特征量。它与 PSF 之间存在投影关系：

$$L(x) = \int h(x,y)\,\mathrm{d}y \tag{3.25}$$

如果 h(x,y)是旋转对称的，投影的方向就可以是任意的。理想的台阶经射线透视成像后在像面上的像将是退化了的具有模糊边缘的台阶像，其中就包含有成像系统的 PSF 信息。设理想直角台阶函数为

$$S_i(x) = \begin{cases} 1 & x > 0 \\ 0 & x < 0 \end{cases} \tag{3.26}$$

那么，台阶样品经射线透射照相模糊像为

$$I_{si}(x) = S_i(x) * h(x,y) \tag{3.27}$$

是理想台阶像与 PSF 的卷积。把式(3.26)代入(3.27)，整理得：

$$\begin{aligned} I_{si}(x) &= S_i(x) * h(x,y) \\ &= \iint S_i(x) * h(\tau - x, y)\,\mathrm{d}y\,\mathrm{d}x \\ &= \int [\int h(x,y)\,\mathrm{d}y]\,\mathrm{d}x \\ &= \int L(x)\,\mathrm{d}x \end{aligned} \tag{3.28}$$

再对(3.28)式微分可得 LSF 为

$$L(x) = \frac{\mathrm{d}}{\mathrm{d}x}[I_{si}(x)] \tag{3.29}$$

根据上述原理，只要测出成像系统中得 LSF,PSF 或 MTF 中得任何一个，就可以求出其它两个。我们对由成像系统得到的台阶模糊像剖面线求导得到成像系统的 LSF；对其做投影得到 PSF；再对 PSF 傅立叶变换，求出 MTF；最后由 MTF 得到成像系统的空间分辨率。

图 3.2 为以上方法过程的示意图。

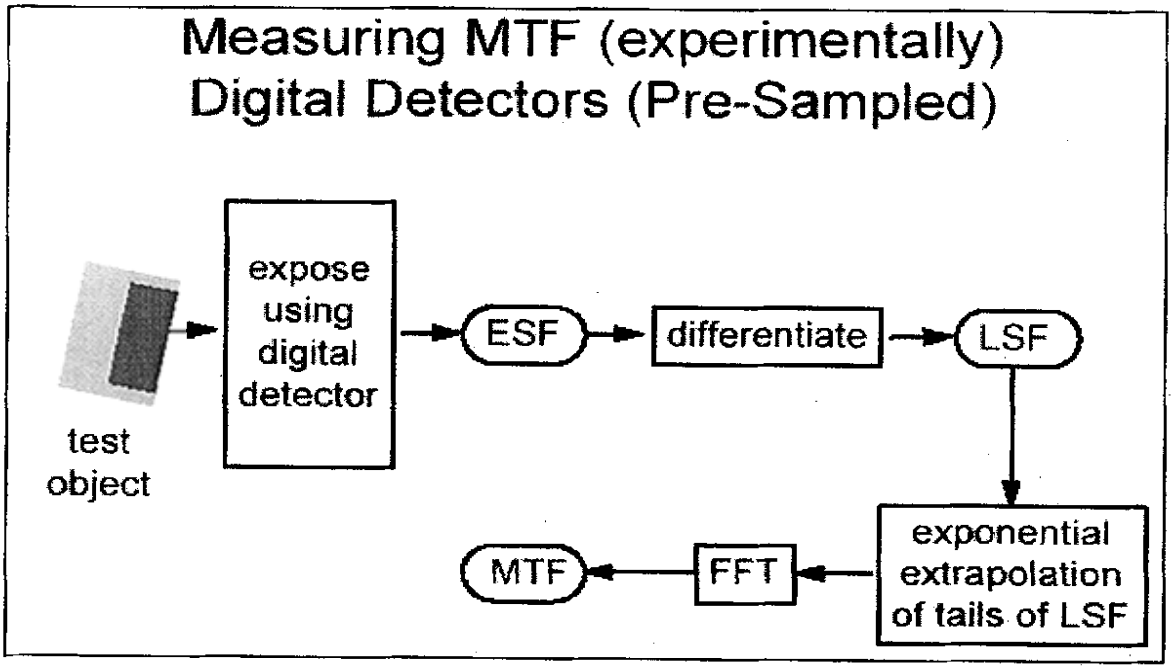


图 3.2　MTF 测量方法示意图

传统意义上，MTF 和 DQE 仅仅是衡量图像接收器的性能，而不是完整的成像系统。因为当我们测量 QE 时，像素单元之间的散射情况并没有加以考虑，系统去噪能力也没有考虑。测量调制传递函数的传统方法是把台阶或缝隙尽可能贴近探测器，然后焦点模糊程度最小化，最后探测器性能独立化。这种方法就过去传统方法上来说是有效的，因为以前的胶片或转换屏相同像素单元之间的相互影响或多或少是等同的，因而图像性能受其它单元影响的程度也是相同的。

由于 MTF 反映的是系统空间分辨率，因此 MTF 大小取决于探测器某些特性，例如探测器材料，厚度等都对空间分辨率有影响。以传统的胶片或转换屏为例，厚度越大，其点扩展函数的半高宽越大，其 MTF 通频能力越差，因此说明胶片或转换屏的空间分辨能力随其厚度的增加而降低[24]。图 3.3 为不同厚度的 CsI:Tl 转换屏的调制传递函数 MTF 的实验结果比较。另外，入射光子能力对 MTF 也有一定影响。我们仍以 CsI:Tl 转换屏为例，图 3.4 为对于不同输入光子能量的 MTF 比较。可以看出，对转换屏而言，其空间分辨能力随入射光子能量的增加而增加[25]。

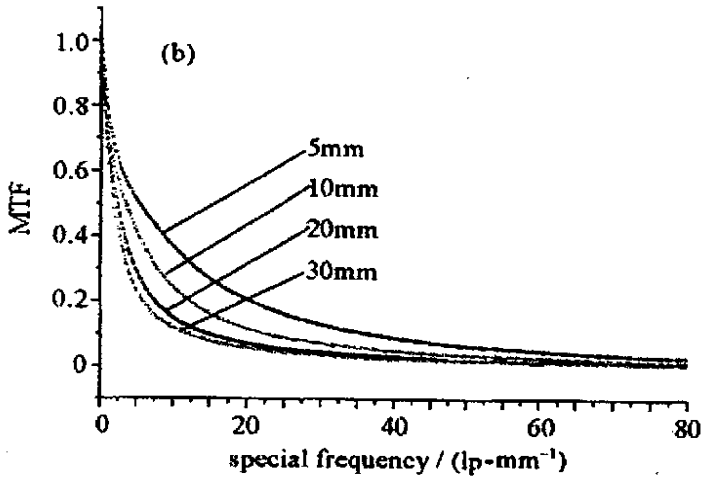


图 3.3　不同厚度转换屏的 MTF 比较

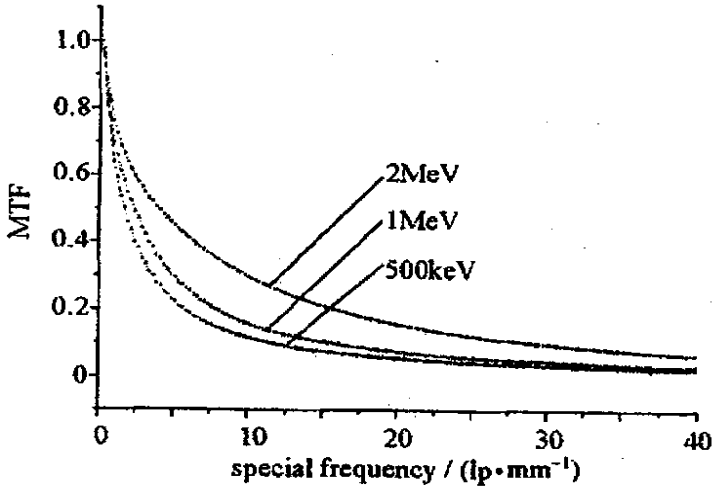


图 3.4 不同入射能量下转换屏的 MTF 比较

在理想情况下，即不考虑像素之间串扰时，探测器空间分辨率主要取决于像素单元的大小。例如，假设探测器像素直径为 d，则探测器调制传递函数 MTF 在频率 $f = d^{-1}$ 为零。若系统有放大功能，这个频率可能将会更高。对于由离散元素组成的数字化成像探测器，空间分辨率主要由采样间距决定。我们假设采样间距为 p，则探测器调制传递函数 MTF 在频率 $f = (2p)^{-1}$（奈奎斯特频率）时为零。换句话说，在图像频率低于奈奎斯特频率 f 时，成像探测器将能够如实分辨。反之，高于奈奎斯特频率 f 时，所成图像将出现模糊现象。在实际数字化成像中，最小采样间隔为像素大小，即 $p = d$，因此奈奎斯特频率 $f = (2d)^{-1}$

在实际成像探测器设计中，当像素大小和采集间隔被确定以后，射线源特性就成为影响探测器系统空间分辨能力的一个重要因素。例如焦斑的不清晰度将会成为 MTF 受限的一个

要原因。另外，在实际探测器中，最终决定系统空间分辨率的是接收器像素之间的噪声影响程度，而并不是像素大小。因此从某种意义上说通过消噪声的方法来改善系统性能比减小像素大小的方法更为有效。

3.5 噪声

消噪声方法有很多。常用的领域平滑，低通滤波等方法虽然都能较好地消除图像中的噪声，但也给图像带来新的模糊，这势必影响MTF的测量精度。根据台阶图像的特点，我们探索用阈值法消除CCD图像中随机分布的白斑噪声，用多条剖线平均法和曲线拟合法消随机分布的其它噪声，取得了较好的效果。

3.5.1 白斑噪声的消除

CCD图像中的白斑噪声呈颗粒状，大小不等，其灰度值比周围像素的灰度值大的多，所以在图像显示为白斑。考虑到整幅图像中白斑噪声像素远小于有用信息像素以及台阶特有的特征(台阶图像在垂直于阶跃边的每一条线上的像素值应相等)，我们采用阈值法消除白斑噪声[29]。

设图像灰度值为 $f(x,y)$，则对于横放台阶，每一行的像素灰度值应基本相等。对一行数据求平均，得到数值 $t(x)$ 为

$$t(x)=\frac{1}{N}\sum_{y=1}^{N}f(x,y) \tag{3.30}$$

用求出的 $t(x)$ 作为阈值，对这一行的每个数据进行检查，凡是大于 $t(x)$ 的数据用 $t(x)$ 代替，否则保留原来的数据。经过这样的处理后，白斑噪声得到有效消除。

3.5.2 涨落噪声和其它噪声的消除

图像中除了白斑噪声外，还含有许多其它噪声，其中大量的涨落噪声也是影响成像系统MTF测量精度的主要因素之一。对于图像中涨落噪声和其它噪声，根据台阶图像的特点，取多条剖线平均和多项式曲线拟合的方法来进行处理。

选取图像中包含台阶界面的 $M\times N$ 区域数据(已消除白斑噪声)，使用多条剖线平均的方法消除涨落噪声。设消除白斑噪声的图像为 $g(x,y)$，则多剖线平均值 $c(x)$ 为

$$c(x)=\frac{1}{N}\sum_{y=1}^{N}g(x,y),x=1,2,3,.....,M \tag{3.31}$$

经过多条剖线平均后，基本上消除了涨落噪声，得到较为光滑的台阶剖面线。但这条曲线中，仍含有一些其它噪声，仍需做进一步处理。经过反复探索发现使用多项式曲线拟合的方法可以较好地消除其它噪声。

经过上述几个步骤的消噪声处理，最后得到台阶剖面线基本上消除了各种噪声，可以进行求MTF的运算了。

3.5.3 测量MTF算法

测量MTF时，通常用差分代替微分作梯度运算。对经过消噪声处理的台阶剖面线进行求梯度运算，即得到LSF为

$$L(x)=\{[f(x)-f(x-1)]^2+[f(x+1)-f(x)]^2\}^{1/2} \tag{3.32}$$

LSF求出后，可以求出系统的PSF。为了简化计算，用LSF近似代替PSF。直接对LSF作傅立叶变换就得出系统的MTF。

图3.5为某原始图像台阶单剖线，可看出台阶剖线很不光滑，有一些大毛刺和许多小毛刺。对此台阶图像消噪声处理和计算MTF结果如图3.6、图3.7和图3.8所示。图3.6中，实线代表消除白斑噪声和涨落噪声后的台阶剖线；虚线代表对台阶剖面线进行曲线拟合的结果。由图3.6可以看出噪声被很好地消除，最后得到的台阶剖线(虚线)已很光滑。图3.7为对台阶剖线作梯度变换的结果，由于很好地消除了噪声，因此求得的LSF比较规则。图3.8为计算得到的MTF函数，从中可读出系统的空间分辨率[30]。

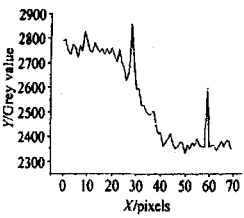


图3.5 原始图像台阶剖线

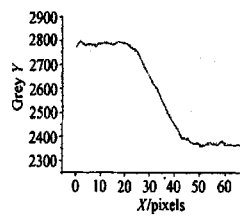


图3.6 消噪声后的台阶剖线

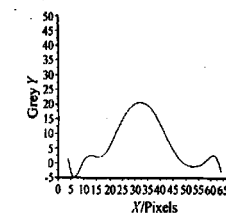


图3.7 消噪声后的LSF

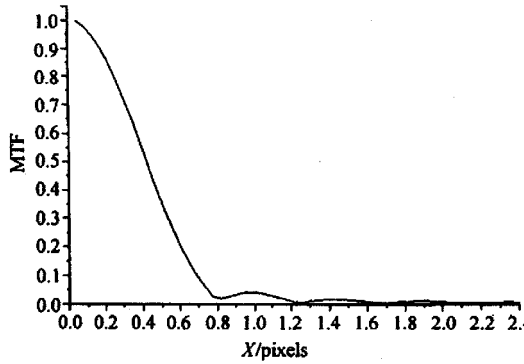


图 3.8 消噪声后的 MTF

噪声对图像的干扰，极大地影响成像系统调制传递函数的测量精度。为了很好地消除噪声而不影响图像中的有用信息，根据台阶图像的特点，采用阈值法、多剖线平均法以及多项式曲线拟合法对图像中的白斑噪声、涨落噪声和其它噪声进行处理。最后对经过消噪声处理的台阶剖线进行计算，求得成像系统的 MTF。反复实践表明，用这种数据处理方法测量成像系统的 MTF 是合适的。

第四章 图像重建

图像重建是数字化成像中的核心工作，采用合适的图像重建算法对最后所成图像质量具有重大的意义。图像重建一般是针对CT成像技术而言的，这是因为CT是显现物体断层图像，所得数据需要经过复杂的算法才能实现投影图像的重建，而DR相对来说算法较为简单，只是基于射线强度的衰减得到数据，然后直接投影重建成像。因此在这里我们主要针对CT成像技术来解释在本成像探测器系统研究中采用的投影图像重建算法。

从投影重建图像的方法很多，常见的有：反投影重建算法、滤波反投影算法、直接富里叶重建算法、迭代重建算法、小波算法。一般商用CT中广泛应用的是滤波反投影算法。因此在系统中我们也采用此算法。

4.1 滤波反投影成像

4.1.1 反投影重建算法

在解释滤波反投影算法之前，我们首先看看直接反投影重建算法。

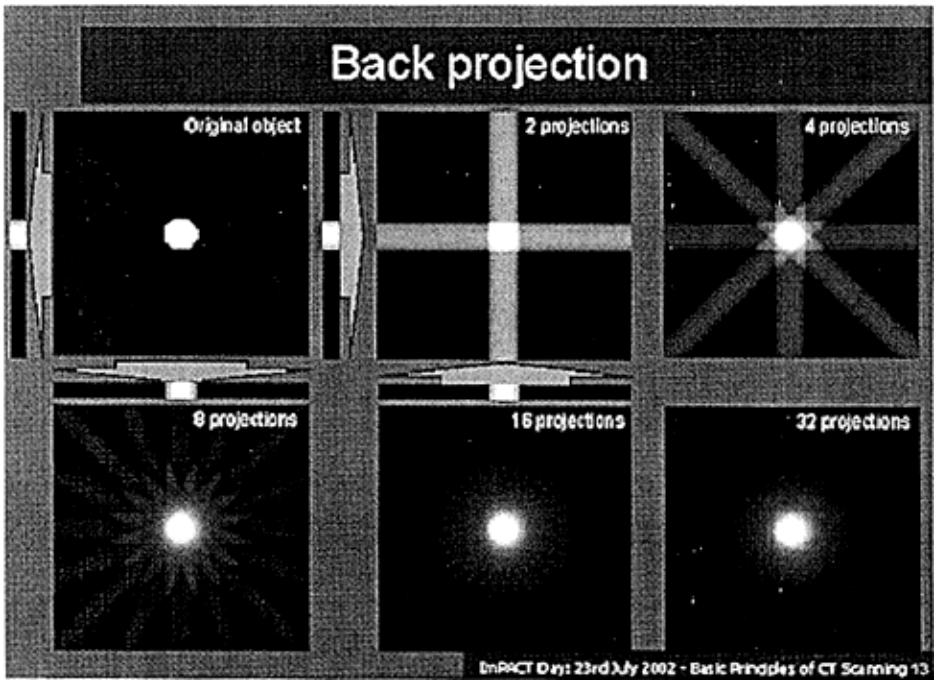


图 4.1 反投影图像重建

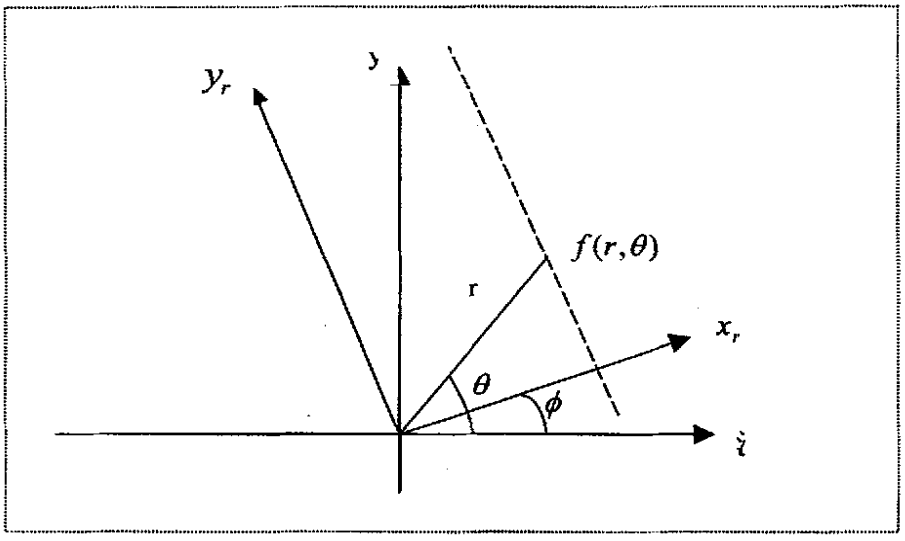


图 4.2 反投影算法坐标示意图

设 A 为物体中一点，p_{iA} 为穿过 A 点的第 i 条射线的投影，则有：

第一次射线的投影为

$$p_{1A} = \int_{L_1} f(x,y)dl$$

第二次射线的投影为

$$p_{2A} = \int_{L_2} f(x,y)dl$$

第 i 次射线的投影为

$$p_{iA} = \int_{L_i} f(x,y)dl \tag{4.1}$$

通过 A 点的所有射线的投影叠加在一起，形成 A 点加强的像。在经典断层成像术中，认为所成的像全部由 A 点密度所贡献（其实还有投影路径上其他点的影响），而这些贡献又与 $\sum_i p_{iA}$ 成正比。那么，就可以把 $\sum_i p_{iA}$ 看作 A 点的密度 $f_A(x,y)$ 的量度。这就是反投影重建图像的思想。

所以，计算机断层成像中反投影重建算法的定义是：

断层平面中某一点的密度值可看作这一平面内所有经过该点射线（反）投影之和（的平均值）整幅重建图像可看作所有方向下的（反）投影累加而成。求和运算导致反投影重建算法的另一名称———累加法。

直接反投影重建算法中，将取自有限物体空间的投影均匀的反投影到射线所及的无限空间的各点之上，包括原先像素值为零的点，各方投影重建后的图像除保留像素点的像外，

还有像素值为1/n的灰雾背景，产生星状尾迹。

建立图4.2所示坐标系。$x-y$为固定坐标，x_r-y_r为旋转坐标，(r,θ)为极坐标。点源$\delta(x,y)$为$x-y$断面中唯一的像点。扫描方式为平移/旋转。X射线方向为沿y_r方向。x_r-y_r坐标系与$x-y$坐标系原点重合，y_r与y夹角为ϕ。不同ϕ代表不同投影角度。ϕ为离散取值$\phi=\phi_n$，相应的投影为

$$p_{\phi_n}=\delta[r\cos(\theta-\phi_n)] \tag{4.2}$$

根据式(1.22)，坐标平面上任一点(x_r,y_r)的图像在坐标系中的表示为。

$$\begin{aligned} f(r,\theta)=f_r(x_r,y_r)&=\frac{1}{N_\phi}\sum_{i=1}^{N_\phi}p_{\phi i}(x_r)\big|_{x_r=r\cos(\theta-\phi_i)} \\ &=\frac{1}{N_\phi}\sum_{i=1}^{N_\phi}p_{\phi i}[r\cos(\theta-\phi_i)] \\ &=\frac{1}{\pi}\sum_{i=1}^{N_\phi}p_{\phi i}[r\cos(\theta-\phi_i)]\Delta\phi \end{aligned} \tag{4.3}$$

其中$\Delta\phi$为投影数$\frac{\pi}{N_\phi}$。

式(4.3)的物理意义是：经过某点的所有射线投影的平均值即为该点的密度值。

若ϕ连续取值，则有

$$f(r,\theta)=\frac{1}{\pi}\int_0^\pi p_\phi[r\cos(\theta-\phi)]d\phi \tag{4.4}$$

$$h(r,\theta)=\frac{1}{\pi}\int_0^\pi \delta[r\cos(\theta-\phi)]d\phi \tag{4.5}$$

如原像为$\mu(x,y)$，则将原像取投影后再按反投影算法得到的重建图像为：

$$f(x,y)=\mu(x,y)**h(x,y) \tag{4.6}$$

即$f(x,y)$为$\mu(x,y)$与$h(x,y)$的二维卷积。

4.1.2 滤波反投影重建算法

上一节我们看到反投影重建算法的缺点是引入星状尾迹，为了尽可能的去除星状尾迹对

成像质量的影响，现在商用CT中广泛采用了滤波反投影的重建算法。这一节中我们就来讨论该算法，这也是我们系统CT成像程序中采用的算法。

4.1.2.1 傅立叶切片定理

滤波反投影算法的基础是富里叶切片定理(The Fourier Slice Theorem)[2]，其内容是：

某图像 $f(x,y)$ 在视角为 θ 时，投影 $P_\theta(x)$ 的一维富里叶变换给出 $f(x,y)$ 的二维富里叶变换 $F(u,v)=\tilde{F}(\rho,\theta)$ 的一个切片。切片与 u 轴相交成 θ 角，且通过坐标原点。即

$$F[P_\theta(x)]=\tilde{F}(\rho,\theta)|_{\theta\text{固定}} \tag{4.7}$$

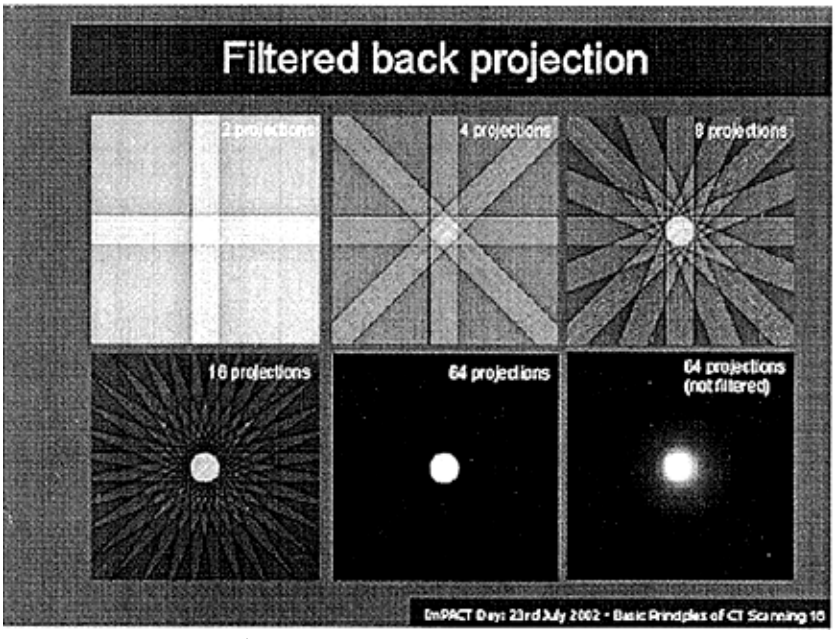


图 4.3 滤波反投影重建

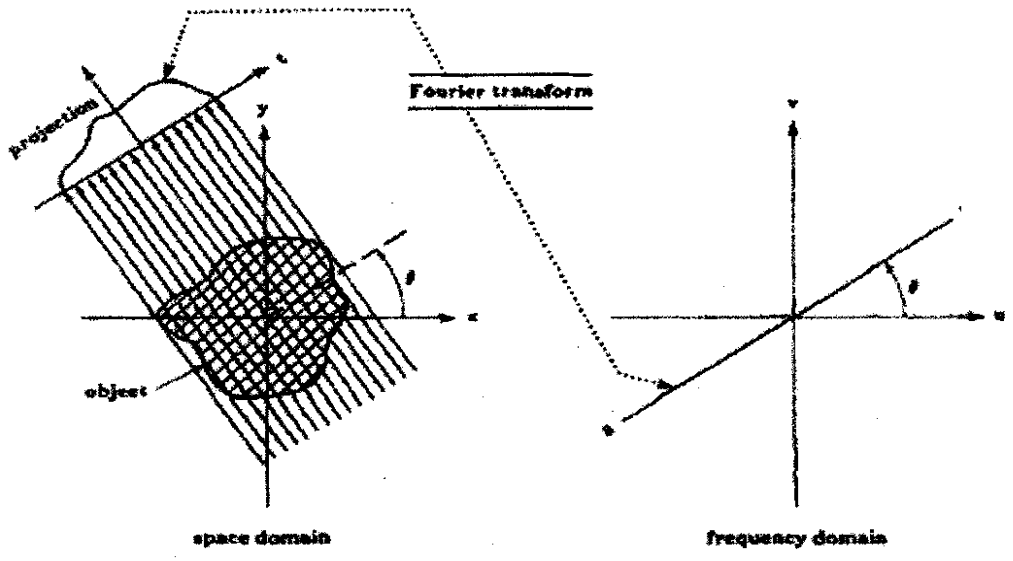


图 4.4 演示富里叶切片定理的坐标系

我们来看看定理的证明：

图像的函数 $f(x,y)$ 的二维富里叶变换是：

$$F(u,v)=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}f(x,y)e^{-j2\pi(ux+vy)}dxdy \tag{4.8}$$

角度 θ 时的投影为 $P_\theta(t)$，它的富里叶变换为：

$$S_\theta(w)=\int_{-\infty}^{+\infty}P_\theta(t)e^{-j2\pi wt}dt \tag{4.9}$$

最简单的例子就是当 $\theta=0$ 的时候。此时频域中 $v=0$，沿此直线的富里叶变换简化为：

$$\begin{aligned}F(u,0)&=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}f(x,y)e^{-j2\pi ux}dxdy\\&=\int_{-\infty}^{+\infty}[\int_{-\infty}^{+\infty}f(x,y)dy]e^{-j2\pi ux}dx\end{aligned} \tag{4.10}$$

由平行射线束的定义

$$P_{\theta=0}(x)=\int_{-\infty}^{+\infty}f(x,y)dy \tag{4.11}$$

用此式替换括号中的部分可得

$$F(u,0)=\int_{-\infty}^{+\infty}P_{\theta=0}(x)e^{-j2\pi ux}dx \tag{4.12}$$

上式等号右边就是 $P_{\theta=0}$ 的一维富里叶变换；所以，我们可以得到垂直投影和图像函数的 2-D 富里叶变换的关系

$$F(u,0)=S_{\theta=0}(u) \tag{4.13}$$

这就是富里叶切片定最简单的例子。

现在来讨论一般情况，坐标系(t,s)相对于坐标系(x,y)旋转了一个角度θ，有

$$\begin{bmatrix} t \\ s \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{4.14}$$

在(t,s)坐标系中沿着 t 为常数的直线的投影

$$P_\theta(t) = \int_{-\infty}^{+\infty} f(t,s)ds \tag{4.15}$$

其富里叶变换为

$$\begin{aligned} S_\theta(w) &= \int_{-\infty}^{+\infty} P_\theta(t)e^{-j2\pi wt}dt \\ &= \int_{-\infty}^{+\infty}[\int_{-\infty}^{+\infty} f(t,s)ds]e^{-j2\pi wt}dt \end{aligned} \tag{4.16}$$

转化到(x,y)坐标系中

$$S_\theta(w) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x,y)e^{-j2\pi w(x\cos\theta+y\sin\theta)}dxdy \tag{4.17}$$

等式右边为空间频率$(u = w\cos\theta, v = w\sin\theta)$的二维富里叶变换，即

$$S_\theta(w) = F(w,\theta) = F(w\cos\theta, w\sin\theta) \tag{4.18}$$

至此证明了富里叶切片定理。

从上面的结果可以看出：只要得到角度$\theta_1,\theta_2,.....\theta_k$的投影及他们的富里叶变换，我们就可得出各个角度的$F(u,v)$。如果$k->\infty$，根据

$$f(x,y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} F(u,v)e^{j2\pi(ux+vy)}dudv \tag{4.19}$$

从$F(u,v)$的值可以得出 u-v 平面内所有点的值。

4.1.2.2 滤波反投影算法

下面讨论基于富里叶切片定理的滤波反投影算法[15]。使用的坐标系如图 4.5。我们只考虑平行射线投影的情况。通过下面的富里叶反变换，$f(x,y)$表示为

$$f(x,y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} F(u,v)e^{j2\pi(ux+vy)}dudv \tag{4.20}$$

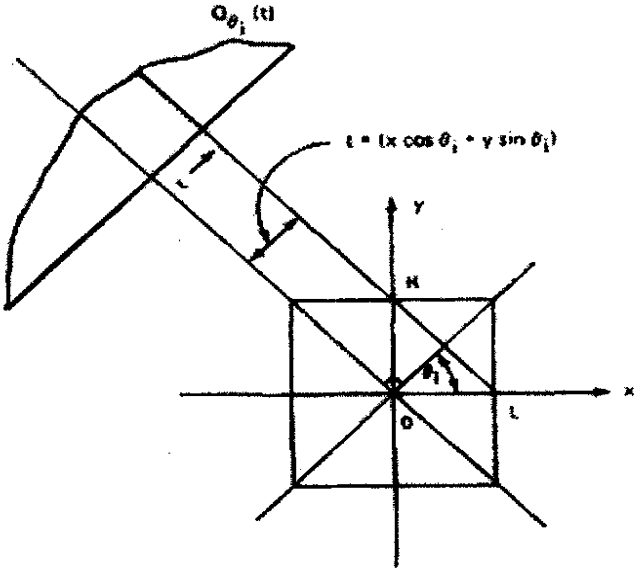


图 4.5 演示滤波反投影算法坐标系

通过变换

$$u = w\cos\theta$$
$$v = w\sin\theta$$

及

$$dudv = wdwd\theta$$

(4.20)式可以在极坐标中写成

$$\begin{aligned} f(x,y) &= \int_0^{2\pi}\int_0^{\infty} F(w,\theta)e^{j2\pi w(x\cos\theta+y\sin\theta)}wdwd\theta \\ &= \int_0^{\pi}\int_0^{\infty} F(w,\theta)e^{j2\pi w(x\cos\theta+y\sin\theta)}wdwd\theta \\ &+ \int_0^{\pi}\int_0^{\infty} F(w,\theta+\pi)e^{j2\pi w(x\cos(\theta+\pi)+y\sin(\theta+\pi))}wdwd\theta \end{aligned} \tag{4.21}$$

然后，利用性质

$$F(w,\theta+\pi) = F(-w,\theta)$$

及关系

$$t = x\cos\theta + y\sin\theta$$

(4.21)式重写为

$$f(x,y) = \int_0^{\pi}[\int_{-\infty}^{+\infty} F(w,\theta)\,|\,w\,|\,e^{j2\pi wt}dw]d\theta$$

$$= \int_0^{\pi} [\int_{-\infty}^{+\infty} S_\theta(w) | w | e^{j2\pi wt} dw] d\theta \tag{4.22}$$

$$= \int_0^{\pi} Q_\theta(x\cos\theta + y\sin\theta) d\theta \tag{4.23}$$

这里

$$Q_\theta(t) = \int_{-\infty}^{+\infty} S_\theta(w) | w | e^{j2\pi wt} dw \tag{4.24}$$

$$= h(t) * P(t,\theta) \tag{4.25}$$

其中

$$h(t) = \tilde{F}^{-1} | w | \quad , \quad P(t,\theta) = \tilde{F}^{-1}[S(w,\theta)]$$

式(4.24)就是一个频率响应由$|w|$给出的滤波函数，因此，$Q_\theta(t)$被称为滤波投影。它的物理意义是投影$P(t,\theta)$经过传递函数为$|w| = \tilde{F}[h(t)]$的滤波器滤波后得到的修正后的投影在满足$t = x\cos\theta + y\sin\theta$时的值。$t = x\cos\theta + y\sin\theta$恰是通过给定点$(t,\theta)$的射线方程。式(4.23)的物理意义是：将过$(t,\theta)$点的所有射线滤波后投影值在$\theta = 0 \sim \pi$范围内的累加，得到$f(x,y)$的一个估计值，此过程就是一个滤波反投影过程。

式(4.23)又称为“滤波反投影方程”，它集中体现了滤波反投影算法的各个步骤。把它分解出来得到图4.6所示的框图。具体说有下述三大步骤：

(1) 把固定视角θ下测得的投影$P(t,\theta)$经过滤波，得到滤波后的投影$Q(t,\theta)$；

(2) 对每个θ，把$Q(t,\theta)$反投射于满足$t = x\cos\theta + y\sin\theta$的射线上的所有各点$(t,\theta)$；

(3) 将步骤(2)中的反投影值对所有$0 < \theta \leq \pi$进行累加（积分），得重建后的图像。

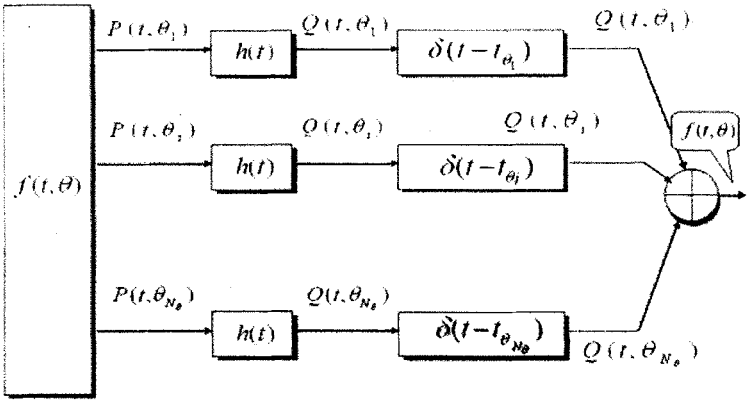


图 4.6 滤波反投影算法步骤图

4.1.3 滤波反投影算法的实现

上面讨论了滤波（卷积）反投影算法，对于这一算法，滤波函数的选取是非常重要的。滤波函数兼指滤波器的系统函数 $H(w)$ 和它的冲击响应 $h(t)$。下面讨论滤波函数的选取和计算机实现时应考虑的一些具体问题。

4.1.3.1 滤波函数的选取

卷积反投影重建算法的实现，理论上要求滤波器的系统函数 $H(w)=|w|$。这是一个频带无限的滤波函数。因为

$$\int_{-\infty}^{\infty}|H(w)|^2dw=\int_{-\infty}^{\infty}|w|^2dw\longrightarrow\infty$$

按佩里-维纳准则，这一理想滤波器是不能实现的。但若能结合具体成像过程考虑，则既可实现，又有足够精度。具体情况是：

（1） 投影数据的高频（空间域）分量幅度很小；

（2） 投影数据是离散采集的；

（3） 存在噪声。

在物体尺寸有限的情况下，投影数据分布在有限范围内。因而严格来说，其频带是无限的。

但若物体的密度在空间变化平稳，则高频分量的幅度确实不大。另外，检测器在接收X射线时，有平均作用，相当于低通滤波。再有，有限的X射线源尺寸，也提供了附加的低通滤波效应。因此，不管怎样，只要采样间隔d足够小，完全有理由认为高频分量很小。投影数据的频带限制在有折叠频率所规定的区间内，即应使$|w|< B=1/2d$。

由于投影数据是离散采集的，表示成序列：

$$\{P(nd,\theta)\}=P(t,\theta)|_{t=nd}=\sum_{-\infty}^{+\infty}P(nd,\theta)\delta(t-nd) \tag{4.26}$$

式中，d为射束平移的步距；θ为某一固定的视角；n为整数；$\{\bullet\}$表示序列。与此对应，滤波函数也取离散形式

$$\{h(nd)\}=h(t)_{t=nd}=\sum_{-\infty}^{+\infty}h(nd)\delta(t-nd) \tag{4.27}$$

这样，重建后的图像密度函数表示为

$$\begin{aligned}f(r,\theta)=f(x,y)&=\int_0^{\pi}P(t,\theta)*h(t)d\theta\\&=\int_0^{\pi}d\theta\int_{-\infty}^{+\infty}P(t',\theta)h(t-t')dt'\\&\approx\sum_{m=0}^{N_\theta}\sum_{n=-N}^{N}P(nd,\theta_m)h(t-nd) \qquad (4.28)\\&=\sum_{m=0}^{N_\theta}[\{P(nd,\theta_m)\}*h(t)] \qquad (4.29)\end{aligned}$$

式中的t不一定为d的整数倍。所以，可以先在离散域中求得投影数据经滤波后的序列值

$$\{\tilde{P}(nd,\theta_m)\}=\{P(nd,\theta_m)\}*h(nd) \tag{4.30}$$

再将滤波后的投影序列内插，得

$$\tilde{P}(t,\theta_m)=\{\tilde{P}(nd,\theta_m)\}*\psi(t) \tag{4.31}$$

采用线性内插，若$nd<t<(n+1)d$，则

$$\tilde{P}(t)=\tilde{P}(nd)+\frac{\tilde{P}[(n+1)d]-\tilde{P}(nd)}{d}(t-nd) \tag{4.32}$$

4.1.3.2 常用滤波函数

(1)R-L 滤波函数

a)系统函数

$$H_{R-L}(w)=|w|\,rect(w/2B) \tag{4.33}$$

式中

$$rect(w/2B)=\begin{cases}1 & |w|<B=1/(2d)\\ 0 & \text{其他}\end{cases}$$

为一矩形窗

b)冲击响应

$$h_{R-L}(t)=2B^2\sin c(2tB)-B^2\sin c^2(tB) \tag{4.34}$$

c)采样序列

$$h_{R-L}(nd)=\begin{cases}1/(4d^2) & ,n=0\\ 0 & n=\text{偶数}\\ -\dfrac{1}{n^2\pi^2d^2} & n=\text{奇数}\end{cases} \tag{4.35}$$

(2)S-L 滤波函数

a)系统函数

$$H_{S-L}(w)=|w|\sin c(w/2B)rect(w/2B)$$

$$=\left|\frac{2B}{\pi}\sin\frac{\pi w}{2B}\right|rect(w/2B) \tag{4.36}$$

b)采样序列

$$h_{S-L}(nd)=\frac{-2}{\pi^2d^2(4n^2-1)}\qquad n=0,\pm1,\pm2,\ldots。 \tag{4.37}$$

4.2 CT 重建算法程序实现

根据以上图像重建算法，我们为基于闪烁光纤的成像探测器编写了一个 CT 图像重建算法程序。

4.2.1 程序流程

图像重建主要由：卷积滤波，射束计算，内插，反投影重建等步骤组成。

总体流程如下图。

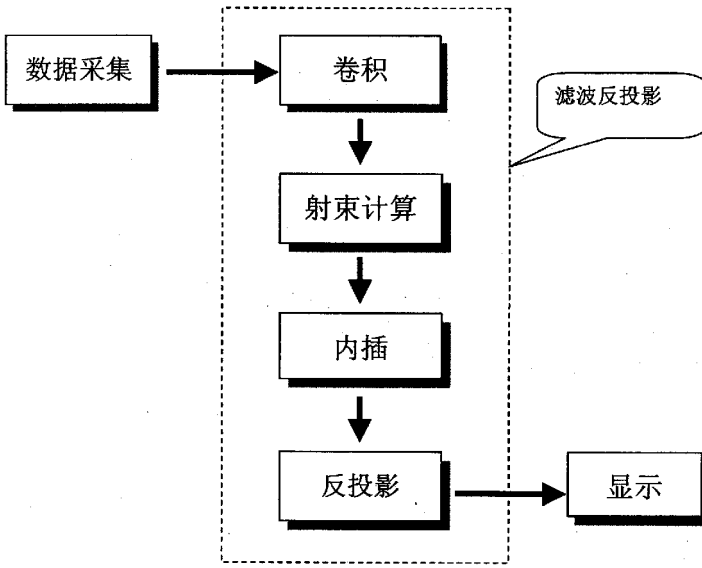


图 4.7 程序总体流程图

程序开发环境为 Microsoft Visual C++ 6.0

整个卷积，反投影过程都被封装在类 FBP 中

图 4.8 为重建时使用的坐标系统。

记平移采样点数为 N_t，平移采样间隔为 d，t 为旋转坐标，t=nd

记角度方向采样点数（投影数）为 N_θ，角度增量 $\Delta\theta = \pi/180$

指定图像画面的像素为 $N \times N$。像素位置记为 (i, j)，i 为 x 方向的坐标，j 为像素在 y 方向的坐标。i, j 的最小值均取 1，左上角的坐标为(1,1)，i 从左至右增加，j 从上自下增加。

4.2.2 程序具体实现

下面就介绍程序具体实现：

4.2.2.1 类 FBP 的实现过程

（1）构造类 FBP 的一个对象。

构造 FPB 对象是通过调用类 FBP 的构造函数实现的

调用时需要传递两个参数<DotsPerLine><Rotations>，也就是 N_t 和 N_θ。构造函数构造一个用来存放滤波函数的 double 类型数组 FilterFunc，然后初始化两个 CObArray 类型的动态数组 PData 和 FPData。PData 存放指向投影数据各行的指针，FPData 存放指向滤波后投影数据各行的指针。

代码：

```
FBP::FBP(int DotsPerLine,int Rotations)
{
    int i;
    nDots=DotsPerLine;
    nRotations=Rotations;
    Zoom=1;

    //initialize the member variable
    FilterFunc=new double[nDots];
    for (i=0;i<nRotations;i++) {
        PData.Add((CObject*)(new double[nDots+2]));
        FPData.Add((CObject*)(new double[nDots]));
    }
}
```

(2)向对象中填充投影数据

在读入数据文件的时候反复调用函数 SetPDataLine 设置投影数据数组 PData 中的每一行数据。

代码：

```
void FBP::SetPDataLine(int LineNumber,double* PDataLine)
{
    int i;
    double* pr=(double*)PData[LineNumber];
    for (i=0;i<=nDots-1;i++) {
        pr[i]=PDataLine[i];
```

```
        }
        pr[nDots]=(pr[0]+pr[1])/2;
        pr[nDots+1]=(pr[nDots-2]+pr[nDots-1])/2;
    }
```

(3) 选择要使用的滤波函数

调用类函数 FBP::SelectFilter(int nFilter)选择要使用的滤波器，nFilter 为滤波器代号，1 为 R-L filter，2 为 S-L filter。调用完成后滤波函数数组 FilterFunc 中就存放好了滤波函数的 N_l 个离散值。

(4) 设置放大倍率

调用 FBP::SetZoomRatio(int ZoomRatio)设置类中数据成员 Zoom 的值。

(5) 滤波

这一部分的工作主要是将投影数据进行卷积运算，得到滤波后的投影值。

设在某一旋转角 θ 时，采得投影数据 $P(t,\theta_m)$。滤波函数为 $h(t)$，滤波反投影

$$\tilde{P}(t,\theta_m)=P(t,\theta_m)*h(t)=\int_{-\infty}^{+\infty}P(t-t')h(t')dt' \tag{4.38}$$

由于数据采集在空间是离散的，经 A/D 变换后幅值也是离散的，故进行离散卷积。令 $t=nd,\theta_m=m\Delta$，对应于 t 取变量 n 位，对应于 θ_m 取变量为 m，于是

$$\tilde{P}(n,m)=P(n,m)*h(n)=\sum_{l=-N_l}^{N_l}P(n-l,m)h(l) \tag{4.39}$$

滤波函数长度为 $2N_l+1$，投影数据长度为 N_l+1，由于卷积过程中还要用到投影数据 $n=-N_l\sim-1$ 及 $n=N_l+1\sim2N_l$ 之间的数据，故需在这两段中填充数据，通常在 $n=-N_l\sim-1$ 中填充 $P^*=\dfrac{P(0)+P(1)}{2}$，在 $n=N_l+1\sim2N_l$ 中填充 $P^*=\dfrac{P(N_l-1)+P(N_l)}{2}$。然后将投影数据序列和滤波函数序列依式(4.39)进行乘加运算，得到卷积和，就是滤波后的投影序列值。

(6) 重建一个像素点

对于图像空间中的一点 (x_i,y_i)，在某一视角下必定对应一 $t_{r,m}=x_i\cos\theta_m+y_i\sin\theta_m$，由于 $t=nd$ 是离散取值的，此点对应的 $t_{r,m}$ 不一定是 nd 的整数倍，即此像素坐标不一定有一射线穿过。故必须用邻近的射线投影进行插值运算才能得到

该点的值，这里我们采用了线性插值。设$t_{r,m}$位于$n_0 d$与$(n_0+1)d$之间，即

$$t_{r,m} = (n_0 + \delta)d \text{，其中} 0 < \delta < 1 \tag{4.40}$$

由式(4.32)

$$\tilde{P}(t_{r,m}, \theta_m) = \tilde{P}_{\theta_m}(n_0 d) + \frac{\tilde{P}_{\theta_m}[(n_0+1)d] - \tilde{P}_{\theta_m}(n_0 d)}{d}(t_{r,m} - n_0 d) \tag{4.41}$$

省去固定的θ_m下标，令d的值为1，上式表示成

$$\tilde{P}(n_0 + \delta) = \tilde{P}(n_0) + \delta[\tilde{P}(n_0+1) - \tilde{P}(n_0)] \tag{4.42}$$

上式就是我们用到的内插公式。因此，我们必须先求得n_0与δ。这在射束计算中完成。射束计算要完成的工作就是得到图像空间中每一像素点对应的n_0与δ。重建图像所用坐标系见图4.8。(i, j)是像素坐标，图像最左上角为$(1,1)$。x, y方向各有N个点。对每一视角，射束从零号射束的位置开始向直径方向增加。由于计算是在(x, y)坐标下进行的，所以先要将(i, j)坐标转化到(x, y)坐标。

由转换公式：

$$\begin{cases} x_i = i - \dfrac{N-1}{2} \\ y_i = -j + \dfrac{N-1}{2} \end{cases} \tag{4.43}$$

可得

$$t = x_i \cos\theta + y_i \sin\theta \tag{4.44}$$

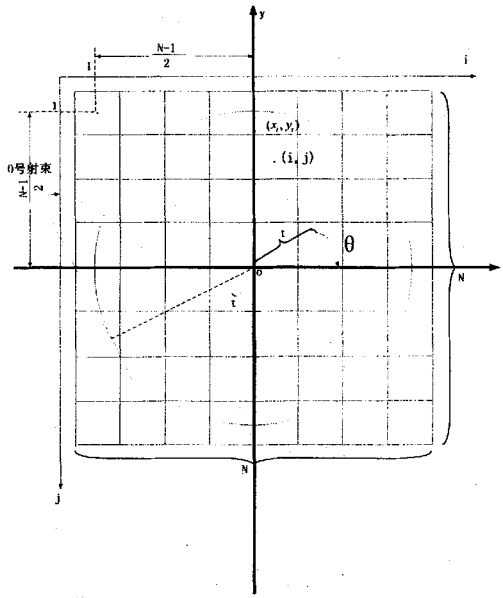


图4.8 重建图像所用坐标系

考虑到图像的缩放，设图像放大z倍，射束计算公式为：

$$\bar{t} = \frac{t}{z} + \frac{N-1}{2} \tag{4.45}$$

它位于射束 $\text{int}(\bar{t})$ 和 $\text{int}(\bar{t})+1$ 之间。

调用内插函数 Interpolate 计算 $\bar{t}$ 处射线的滤波投影值。然后将所有角度的滤波投影值循环求和，就得到点 (i, j) 的像素值，类中函数 DotReconstruct(double xi, double yi)实现了对一个点的重建，重复调用此函数，就可以重建图像中的每一个点。

4.2.2.2 程序总体实现

1)程序代码结构

程序采用 MFC 单 文档/视图 结构，主要有以下几个类

a) 类 CCTImageDoc:程序的文档类，主要用来存放程序数据(投影，反投影,投影数，旋转角度数)，进行数据文件的读入和数据初始化。

b) 类 CCTImageView:程序的视图类，主要进行和用户的交互、窗口的绘制。

c) 类 FBP:滤波反投影的封装库。

2)数据文件的格式和读入，数据的初始化。

该部分功能是在 CCTImageDoc 的序列化函数 Serialize 中实现的。

程序数据以文本方式存放，文件格式为：

旋转角度数

单个角度的射线数

初始射线强度 I_0

保留数字(任意)

[角度编号]

射线号 射线强度

⋮

⋮

射线号 射线强度

[角度编号]

⋮

⋮

程序按行读入数据，并同时计算投影值，读完一个角度的数据后就调用 FBP 类函数 SetPDataLine 设置一个角度的投影值。

4.2.3 一些图像重建结果

a)　　程序界面

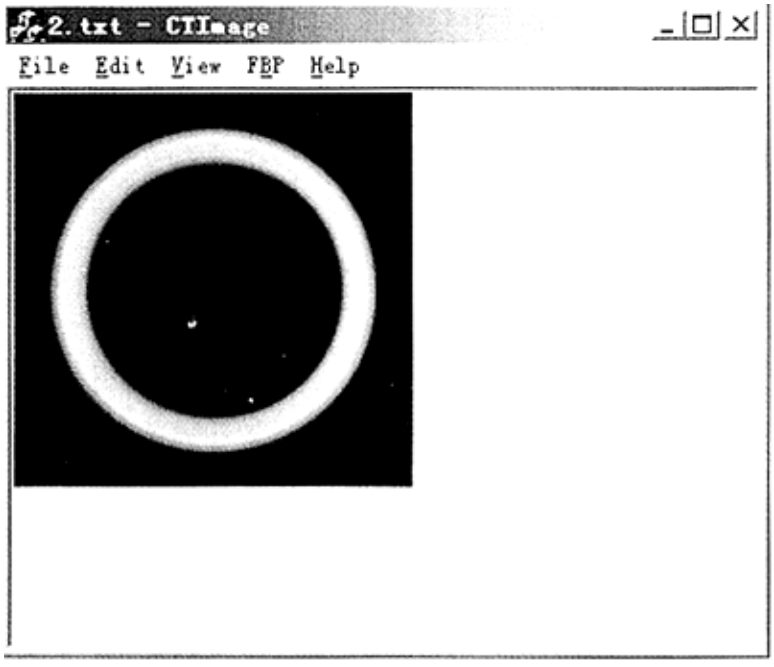
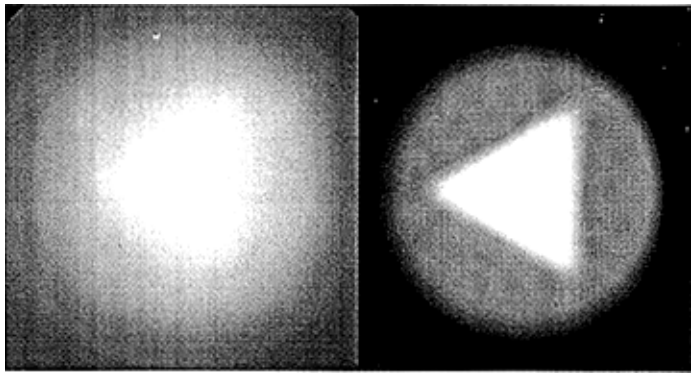


图 4.9 程序界面图

b) 滤波前与滤波后的图像对比



滤波前　　滤波后

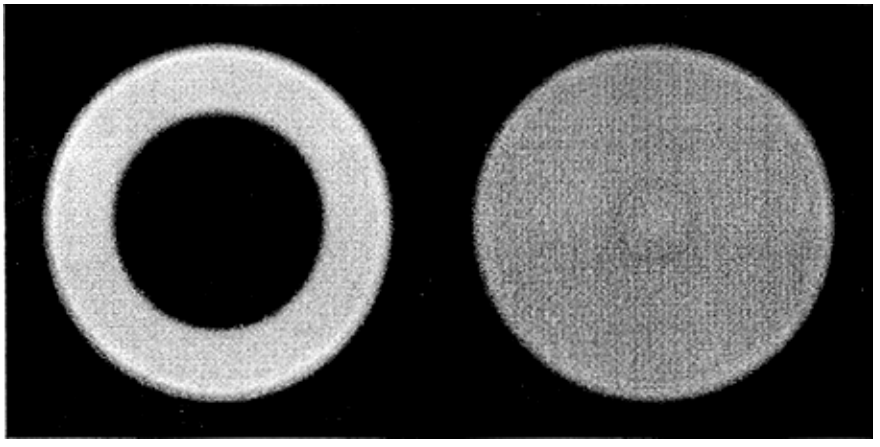
图 4.10 滤波前后对比图

c) 程序模拟数据成像

下面两图是用程序模拟得出的数据进行成像的结果

图 4.11(a)为一圆环截面，单角度射线数 38，采样角度数 90。

图 4.11(b)为一圆截面，单角度射线数 38，采样角度数 90。



(a)　　(b)

图 4.11 模拟数据成像图

d) GEANT4 模拟数据成像

图 4.12 是用 GEANT4 模拟的圆环柱形物体的横截面造影，一个直径很小的圆柱体位于物体中轴线上，在截面图上呈现为一点。

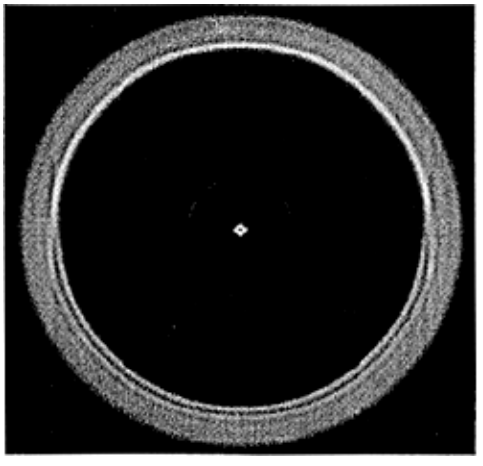


图 4.12 模拟数据成像

e)实验数据成像

图 4.13 是实际用 X 射线实验得到的结果，上面的图片为实物，下面的图片为断层截面的 X 光造影。

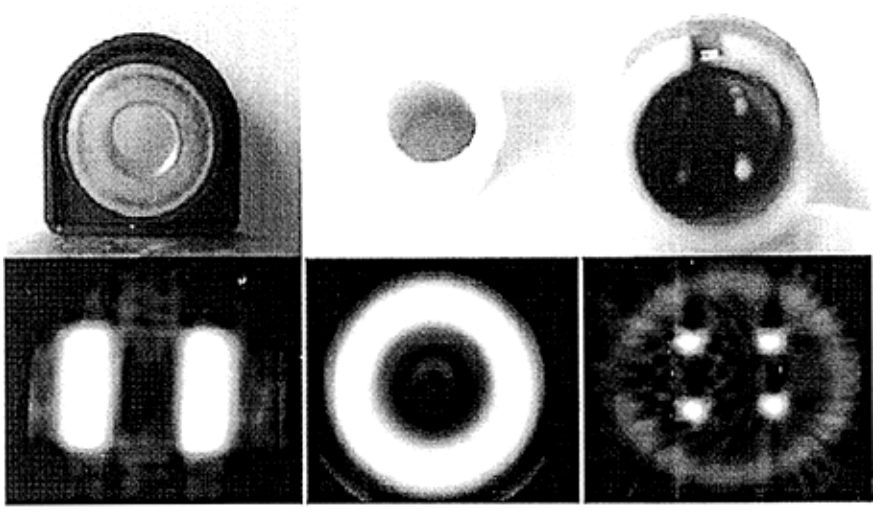


图 4.13 实验数据成像

4.3 图像重建的总结与展望

这里我们初步实现了用滤波反投影的方法重建图像，而且也得到了比较好的结果。现在只考虑了平行射线束的成像，而且暂未考虑对投影数据的修正，噪声的去除等。一些实用CT中用到的扇形射线束成像这里还未实现，近几年又出现了比较先进的螺旋CT，可以连续的采集所有截面数据，而且扫描速度很快，可以较好的重建3D图像。这些都有待以后继续研究。

第五章 基于闪烁光纤的高能X射线成像可行性研究

传统的高能X射线成像探测器采用的是胶片或转换屏。但由于在高能下，其吸收效率不能满足要求，这时需要通过增加胶片或转换屏厚度来增强吸收效率，但增加其厚度的同时会导致图像的空间分辨率下降。这从第三章的图3.3的表征空间分辨率的MTF图中可以看到。从第三章我们知道，吸收效率 Ta=吸收光子能量/入射光子能量。同时，探测器信噪比是由入射光子与探测器作用次数，即被探测的光子数目决定的[16]。因此一个简单的探测器信噪比表达式为：

$$SNR=\frac{\bar{I}}{\sigma_x}=\sqrt{TaN} \qquad (5.1)$$

式中，$\bar{I}$ 为平均信号，σ_x 为背景噪声的标准分布，用信号浮动的算术平方根来表示；N为与探测器碰撞的入射光子数目。为了增加吸收效率 Ta，探测器厚度必须增加，但由于增加厚度后导致横向散射光子的增加，因此极大的降低了空间分辨率。而探测器厚度太小使得吸收效率弱，最终导致信噪比太小。因此在高能下采用传统成像探测器，存在着高空间分辨率与高吸收效率这两者之间的矛盾。所以，在高能下采用这种成像探测器不是一种很好的选择。这使得我们必须寻找一种更为理想的高能X射线成像探测器。

闪烁光纤阵列探测器是一种离散型探测器，它与传统的胶片或转换屏在结构上有较大的区别。它是由许多作为象素单元的单根闪烁光纤整齐排列而成的，即每个象素单元都具有独立的闪烁光纤结构。每根闪烁光纤独立承担产生可见光和光导的作用，并受每根光纤 Clad 包层和 EMA 的作用，其内部产生的可见光光子沿着单根光纤向两端传输。因此从理论上说，闪烁光纤成像探测器空间分辨率是由每个光纤大小所决定的。换言之，若仅仅是通过增加光纤长度来增加吸收效率 Ta，并不会影响探测器的空间分辨率。图5.1比较了传统成像探测器与闪烁光纤阵列在结构和成像过程上的不同。从图中可以看到，在闪烁光纤中，只有一部分产生的可见光光子通过全反射沿着光纤向两端传输，其余的几乎都被 EMA 层吸收。

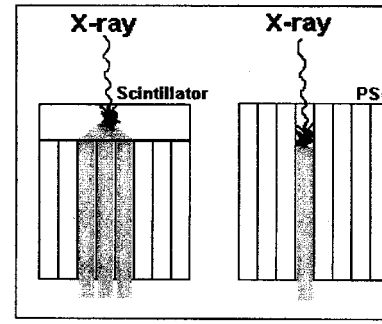


图 5.1 传统闪烁体与闪烁光纤阵列作为成像探测器时的不同设计

通过以上对闪烁光纤阵列与传统转换屏之间的比较，我们设想是否可以将闪烁光纤阵列运用在高能X射线（<20MeV）成像中。为了能够定量的证明此设想的可行性，我们在进行实验验证之前必须进行计算机模拟计算，对闪烁光纤阵列成像探测器的一系列参数，包括SNR,DQE,MTF等做定量的评估，从而更好的对下一步的实验工作进行指导。

5.1 Geant4 模拟工具简介

计算机模拟实验在物理学研究中占着越来越重要的地位。蒙特卡罗方法是计算机模拟采用的常用方法，它是一种随机模拟方法或统计实验方法。它通过不断产生随机数序列来模拟过程。自然界中有的过程本身就是随机的过程，如粒子的衰变过程，粒子在介质中的输运过程等。当然，蒙特卡罗方法也可以借助概率模型来解决不直接具有随机性的确定性问题。因此蒙特卡罗的基本思想就是根据待求随机问题的变化规律，根据物理现象本身的统计规律，或者人为的构造出一个合适的概率模型，依照该模型进行大量的统计实验，使它的某些统计参量正好是待求问题的解[8]。

Geant4是由欧洲核物理联合研究中心（即CERN），开发的一个基于蒙特卡罗方法的包容大量探测器描述和模拟工具的软件系统[4]。随着高能物理(HEP)（high energy physics）实验广度和复杂度的不断增加，模拟工作日益成为探测器设计、优化，实验数据分析、重建等必不可少的手段，Geant4正是从事这方面研究工作的物理学家们的有力工具。

Geant4在HEP中主要应用在两个方面：(1) 对穿越实验装置的粒子径迹进行跟踪，并

模拟粒子与探测器的互作用；（2）给出实验装置的几何描述和粒子径迹的图形表述[6]。

用户在应用 Geant4 系统进行实验装置模拟时必须从以下四个方面来开展工作：

（1）建立一个完整的 C++编译环境；

（2）产生和提供数据以描述实验装置；

（3）了解如何控制程序的执行，并提供相应数据；

（4）编写相关的用户例程，编译运行。

而在所有这些工作中，最重要的就是在程序中建立起虚拟设计的实验装置——我们运用 Geant4 正是要设计、优化、评估闪烁光纤成像探测器，以给出最佳的方案和准确的性能。

5.2 Geant4 模型的建立

在模拟计算时，Geant4 提供了很多的控制参数，其中用户在模拟时常用的参数以 mac，即宏文件的形式提供。在 Geant4 系统中与 X 光子有关的物理过程主要是电子对效应（electron-positron pair conversion），康普顿散射（Compton scattering），光电效应（photo electric effect），重核的光子裂变（photo fission of heavy elements）以及瑞利效应（Rayleigh effect）[7]。考虑到 X 射线与物质相互作用的机制，在以碳，氢，氧为主要成分的材料中，重核裂变和瑞利效应不予考虑（Geant4 的默认设置）。

另外，模拟跟踪的过程中，次级粒子的能量阈值选择也会影响计算结果。这些参数可以通过在类中定义的 CUTVALUE 值来设置，我们的模拟计算中只需要考虑光子和电子的能量阈值就可以了。在 Gean4 中，CUTVALUE 的值在默认条件下以距离(Cut in range)的形式出现，它与能量阈值(Cut in energy)是一一对应的关系，表示在某个能量阈值下，射线粒子在物质中所经过的距离[4]。在此模拟中我们定义 X 射线的 CUTVALUE＝1mm，该值在空气和光纤中对应的能量阈值表如下所示：

表 5.1　X 粒子与 e 在空气与聚苯乙烯中的距离阈值与能量阈值的对照关系

	距离阈值 Cut in range / mm		能量阈值 Cut in energy / keV	
	X	e	X	e
空气 Air	1	1	0.99	0.99
聚苯乙烯 Polystyrene	1	1	2.37	351

Geant4 中模拟粒子与探测器的相互作用是通过蒙特卡洛方法来实现的，这必然地要求我们在模拟计算过程中考虑适当的事例数目[10]。事例过少，则随机效应过于显著而不能真实地反映结果；事例过多，则会消耗宝贵的计算资源，得不偿失。因此，我们用不同的随机

种子研究了随机数效应对不同事例数目计算结果的影响。

可以想见，事例数越大则随机效应越小。计算中事例数选取在 100—1000 之间时结果涨落幅度最高达 200%，显然可靠性很差。我们考察了 10^4，10^5 和 10^6 个事例的情况，发现使用 100000 个事例进行计算比较合适（10^6 个事例固然会减小随机影响，但是将增加 10 倍的计算量，不利于计算资源的有效利用）。

图 5.2 中比较了 10^4 个事例和 10^5 个事例累计 1000 组的计算结果。以 ELOSS 为纵坐标 Y(单位：GeV)，运算次数为横坐标 X(无物理单位)，用一次函数 $Y(X) = A_0 + A_1 \times X$ 来进行回归拟合，结果如下：

对于 10^4 个事例的情况：A_0 = 0.75649E−05，A_1 = −0.10654E−09，χ^2 = 0.8488E−07；

对于 10^5 个事例的情况：A_0 = 0.76162E−05，　A_1 = −0.37586E−10，χ^2 = 0.7676E−08

由于拟合参数使用最小二乘法 (Least squares fit method)，χ^2 值代表了拟合程度的好坏。从上面参数可以看出，使用 10^4 个事例时斜率已基本接近于 0，χ^2 值也较好，使用 10^5 个事例时斜率减小了 60%，χ^2 值减小了一个数量级，达到 10^{-8}。在实际模拟计算中我们选取 10^5 个事例作为标准。

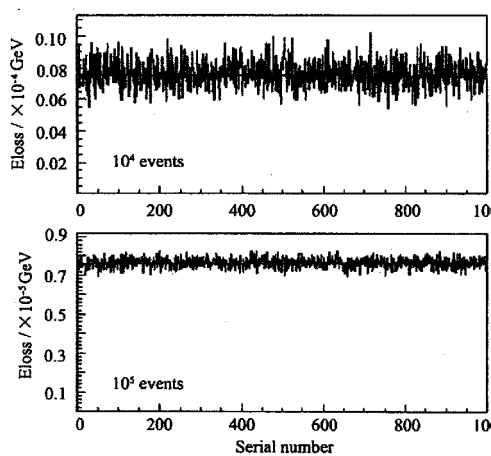


图 5.2　合适事例数的选择

在 Geant4 系统中没有关于闪烁材料发光机制的模拟，因此，在我们的模拟研究中将无法通过 Geant4 直接给出探测器的光输出量，而只能得到射线粒子在探测器中沉积的能量值。虽然闪烁材料发光机制比较复杂，难以对具体的作用过程进行模拟计算，但是闪烁体的光输

出S满足一定的统计规律：

$$S = \frac{n_{光子}}{E} \tag{5.2}$$

式中E为核辐射在闪烁体中损失的能量，$n_{光子}$表示该闪烁过程产生的光子数目。

闪烁体的发光效率表示其将所吸收的核辐射能量转变为光能的性能，又称能量转换效率，用$C_{发光}$来标记。$C_{发光}$与S的关系为：

$$C_{发光} = \frac{n_{光子}\overline{h\nu}}{E} = S\overline{h\nu} \tag{5.3}$$

式中$\overline{h\nu}$是闪烁光子的平均能量。

对于特定的材料、加工工艺和工作温度，闪烁体具有一定的发光效率，也就说明辐射能量的沉积与光输出强度（或闪烁光子数目）存在一定的线性比例关系。因此，在模拟计算中可以用能量沉积量来代替光输出量，这样就可以把能量沉积数据直接作为闪烁光纤的投影数据来进行图像重建，从而简化了模拟测量数据的输出。

5.2.1 单根闪烁光纤的模拟模型

我们首先建立了单根闪烁光纤的Geant4模型，对单根光纤在高能X射线下的辐照特性和光纤阵列在高能成像时的设计，优化进行模拟研究。

在单根闪烁光纤的模拟研究中，主要是针对单根光纤在高能X射线下的吸收效率，探测效率以及最佳长度等参数与入射能量之间的关系，从而寻求出诸如高能下的一个最佳入射能量范围，最优化光纤长度等设计思路，对今后的实验工作起决定指导的作用。

由于我们关心的是单根光纤的辐照特性，根据Geant4模拟系统的优势，在此我们可以特别地研究单根光纤作为探测敏感单元时的能量特性。在设计探测器的几何结构时，我们注重了简单的原则，其空间结构如图5.3。

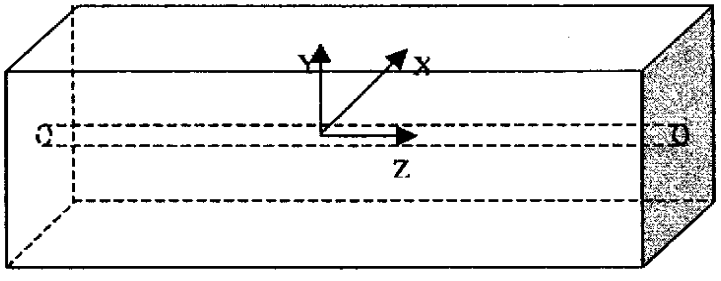


图5.3 单根闪烁光纤模拟模型

图中光纤的几何中心与探测器空间（长方体）的几何中心相重叠，解析坐标如图中所示，Z轴沿光纤中心轴向，X和Y则是沿光纤径向，为直角坐标系。我们在模拟时为了模型简单

起见，选取长方体空间的材料（介质）为空气，基本不与入射高能粒子作用[9]。对于射线源的位置选取我们仍然基于最简单的想法，采用沿 Z 轴方向对准光纤探测器中心位置，由于假设空间介质空气不会对 X 射线粒子产生作用，因此在距离的设定方面可以随意选取。模拟中采用的闪烁光纤类型是 BCF-20，其特性是产生 2.5eV 的可见光子，每 1MeV 的能量沉积能够产生 8000 个可见光子，发射谱的波长峰值为 492nm，衰减时间 2.7ns。光纤直径设为 1mm。Clad 包层厚度为光纤核心直径的 3%。其它密度，折射率等均按照光纤实际参数设置。

图 5.4 为单根闪烁光纤的 Geant4 模型以及与 X 射线作用时的模拟过程截图。当放大图中光纤时，我们可以清晰看到闪烁光纤中基于全反射原理进行传输的可见光子。另外，若是低能 X 射线入射时，X 光子可能与介质中的空气发生作用，但这一点在高能条件下，可以忽略。

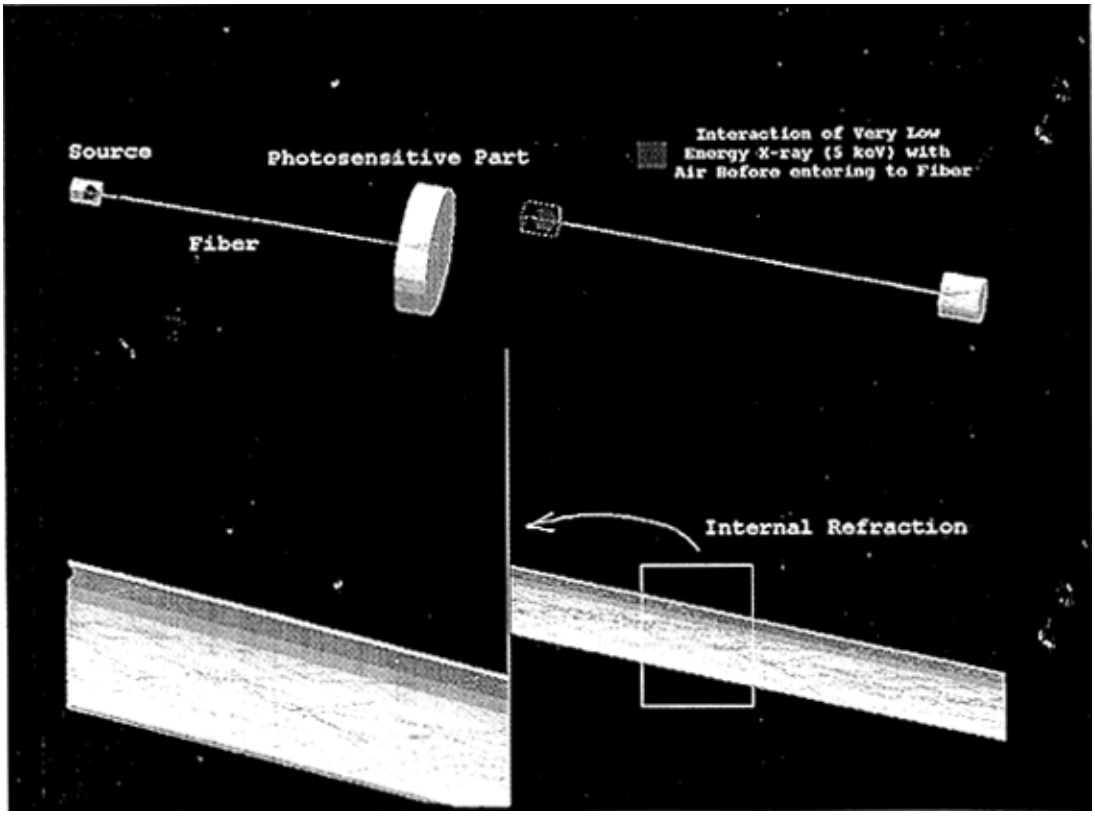


图 5.4 单根闪烁光纤的 Geant4 模型以及与 X 射线作用时的模拟过程截图

5.2.2 闪烁光纤阵列的模拟模型

为了能够定量的评估二维闪烁光纤阵列成像探测器所成图像的质量，在 Geant4 中，我

们建立了一个 25x30 的闪烁光纤阵列模型。其中闪烁光纤型号和参数与单根光纤模型一致。设阵列中心处为空间坐标零点（0,0,0）。为检测成像性能，必须建立一个被测物体模型，因此我们创建了五个不同厚度的金属铜块模型，形成阶梯状，放在射线源与光纤阵列之间。在射线源方面，我们设定射线源为点源，向空间 4π 立体角随机放出高能 X 射线。为检测出闪烁光纤输出端的可见光子数目，我们还设立一个输出可见光子的数目计数器，将其紧贴在闪烁光纤输出端，取名为 PMT。图 5.5 为此闪烁光纤阵列成像探测器的 Geant4 模型。

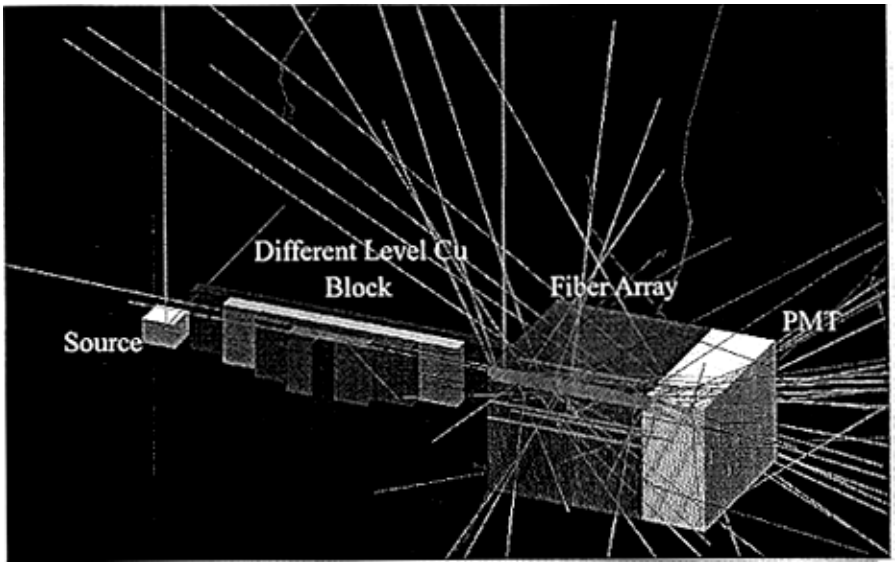


图 5.5 闪烁光纤阵列的 Geant4 模型以及与 X 射线作用时的模拟过程截图

5.3 模拟结果及分析

5.3.1 有关单根闪烁光纤模型的模拟结果

首先，我们试图了解不同入射能量条件下，光纤长度与光纤的能量沉积效率，即吸收效率(Ta)之间的关系如何。这对我们在实验时决定采用多长的光纤具有指导意义。通过对几个不同入射能量下的模拟，我们得到：当闪烁光纤长度增加到 10—12cm 时，能量沉积效率达到一个饱和值。如图 5.6 所示。经过调研，这个结论验证了多年前国外相关领域的工作者做的一些实验结论[18]。

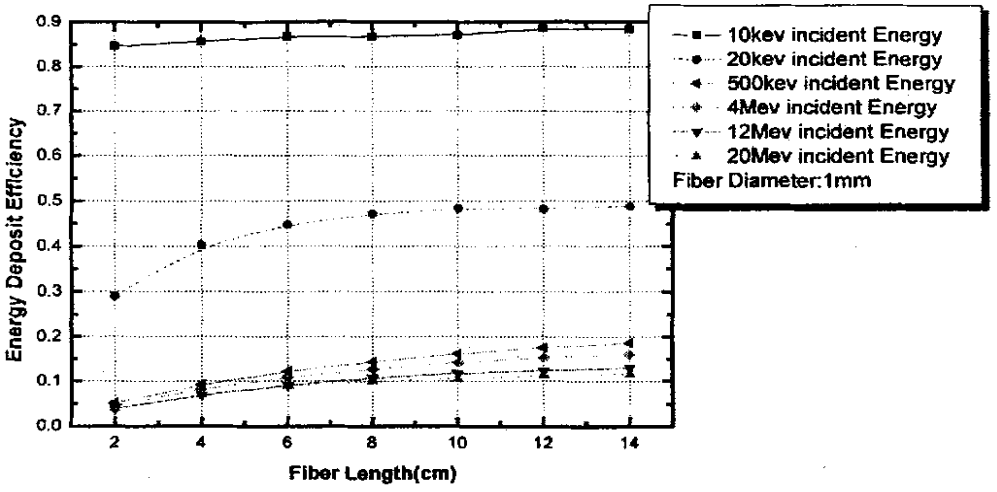


图 5.6 不同入射能量下闪烁光纤长度与能量沉积效率之间的关系

根据此图我们知道，当在高能 X 射线入射下，并不能一味增加光纤长度来提高吸收效率。当闪烁光纤增加到约 10—12cm 左右时，此时的吸收效率趋于饱和，超过这个长度值后对光纤吸收效率贡献不大。并且闪烁光纤越长，对阵列来说越不利，因为光纤越长，阵列中光纤与光纤之间的串扰噪声越大，对空间分辨率越不利。另外把很长的若干根光纤捆绑在一起形成光纤阵列在工艺上也很难做到。因此，在以下的模拟计算中，我们都把闪烁光纤长度设为 10cm。

对于成像探测器来说，量子效率是一个重要的性能衡量标准。在模拟计算中，我们用被闪烁光纤探测到的粒子数与入射粒子数之间的比值来表达量子效率。图 5.7 为 X 射线入射能量与闪烁光纤量子效率之间的关系[12]。图中可以看出，随入射能量的增大，探测器量子效率将降低。当入射能量达到 20MeV 时，量子效率已降到 15%左右。这是由于入射能量越高，穿越探测器粒子数目就越多。因此在高能 X 射线成像下，一般对射线源的剂量要求比较高，否则被探测到的粒子数目就会偏少，最终导致图像的信噪比降低。

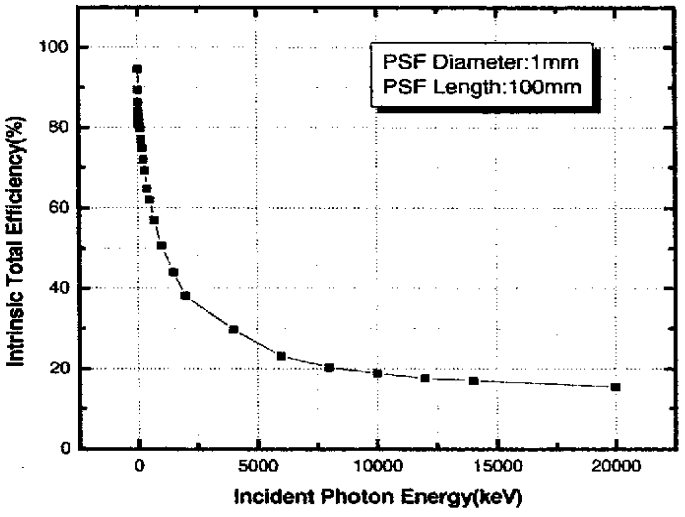


图 5.7 入射能量与闪烁光纤量子效率之间的关系

从第三章我们知道闪烁光纤探测效率(DE)由它的吸收效率(Ta)，转换效率(Tc)，传输效率(Tt)三者决定的，因此要想了解闪烁光纤在不同入射能量下的探测效率，必须对 Ta，Tc，Tt 分别进行模拟计算，然后三者的乘积就得到探测效率 DE。吸收效率 Ta 是通过计算光纤与一定数量的入射光子发生作用后产生的能量沉积，而 Tc 和 Tt 都是常数，它由光纤本身特性决定的，与入射粒子能量无关。例如在模拟中采用的 BCF-20 这种光纤，转换效率 Tc 为每 1MeV 能量沉积产生 8000 个可见光子。传输效率(Tt)则由闪烁光纤材料的折射率以及衰减长度等因素决定的，其大小总是小于光纤的可见光子俘获效率(Ttrapp)，原因在于在可见光子传输过程中有相当一部分光子被吸收或被衰减，实际上，只有一小部分可见光子传到输出端[14]。在本模型中我们设置的 clad 包层和 core 折射率分别为：$\mathbf{n_{core}} = 1.6$ 和 $\mathbf{n_{clad}} = 1.49$。因此，它的俘获效率

$$\mathbf{T_{trapp}} = [1-(\mathbf{n_{clad}} / \mathbf{n_{core}})] / 2 = 3.4\% \qquad (5.4)$$

由于 Tc 和 Tt 为两个常数，因此探测效率 DE 与入射能量之间的关系主要由吸收效率 Ta 决定，而 Tc*Tt 则被定义为闪烁光纤的灵敏度。这在模拟结果中也得到了验证。图 5.8 为吸收效率 Ta 与入射能量之间的关系图，图 5.9 为探测效率 DE 与入射能量之间的关系图，可以看出两者在形状上是基本一致的。

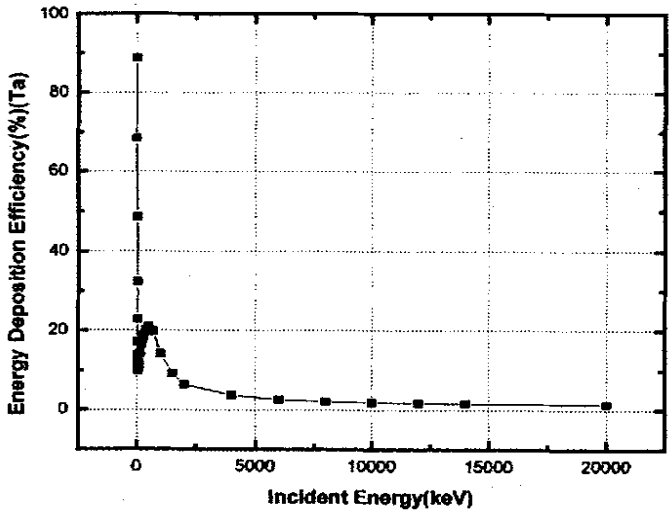


图 5.8 闪烁光纤吸收效率 Ta 与入射能量之间的关系

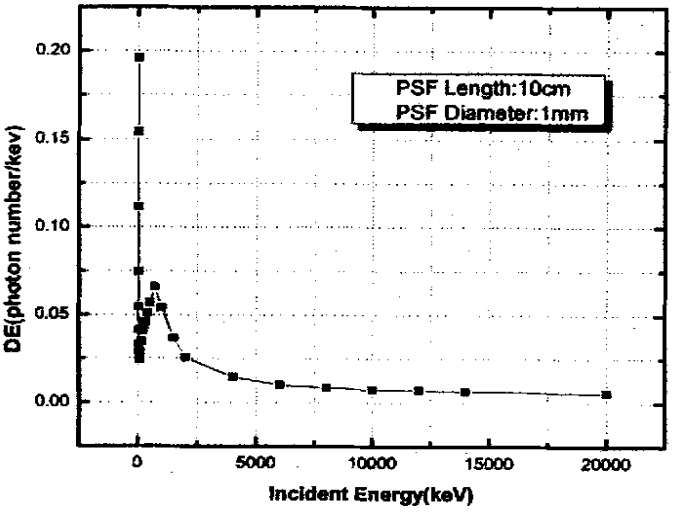


图 5.9 闪烁光纤探测效率 DE 与入射能量之间的关系

结合这两幅图，我们可以看出，由于在低能区域，入射光子与闪烁光纤之间的作用以光电效应为主，能量沉积效率较大，而在 60keV 之后，光电效应次数减少，变为康普敦散射为主，能量沉积效率减小。因此在低能区域，探测效率较高，随着能量的增加，探测效率显著下降。在 2MeV 以上高能区，探测效率变化不大。另外，在 500keV 左右，由于受到闪烁光纤自身材料吸收系数的影响，探测效率有个突然增大，然后再减小的过程。图 5.10a 为闪烁光纤的主要材料聚苯乙烯(Polystyrene)的吸收系数曲线图(图中 μ 和 μ_{en} 分别为衰减系数和

和吸收系数)。为了清晰的进行比较，我们把图5.9的横坐标范围缩小到5000keV，如图5.10b所示。从图5.10a,b两个图可以看出，由于在500keV左右能量处，聚苯乙烯的吸收系数出现一个小峰值，这导致了闪烁光纤探测效率DE在相应位置处也有个峰值。

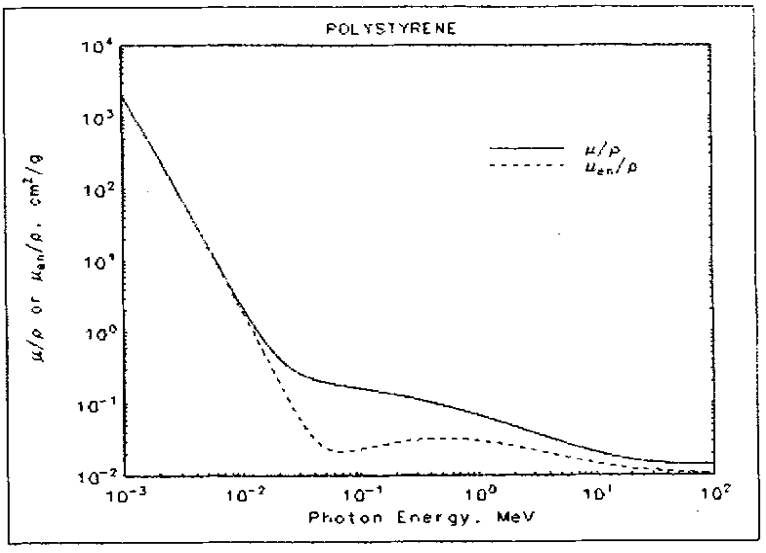


图5.10a 聚苯乙烯(Polystyrene)的吸收系数曲线图

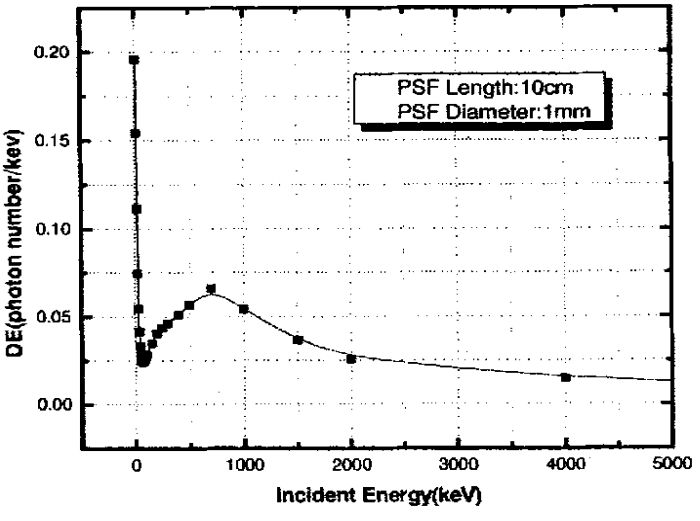


图5.10b 闪烁光纤探测效率DE与入射能量之间的关系

图5.11为闪烁光纤的输出可见光子数目，即闪烁光纤响应，与光纤能量沉积之间的关系，拟合后它是一条过零点直线，而这条直线的斜率就是Ta*Tc，即为闪烁光纤的灵敏度。在我们建立的模型中，闪烁光纤灵敏度为0.25个/keV，即250个/MeV；而根据BCF-20理论参数得：

Tc*Tt=8000个/MeV*Ttrapp=8000*3.4%=272个/MeV

可以看到模拟值略小于理论值。

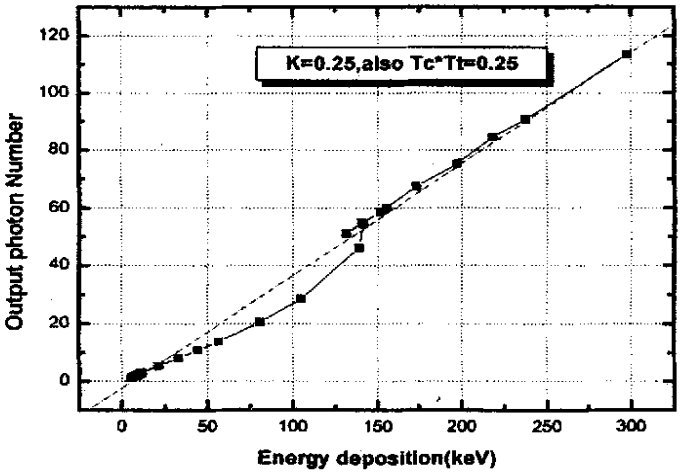


图 5.11 闪烁光纤输出光子数目与能量沉积之间的关系

通过对单根闪烁光纤部分辐照特性的模拟可知，就探测效率而言，闪烁光纤在高能条件下探测效率较低，这就要求射线源有较大的强度，这样才能有足够多的粒子被探测到，保持系统具有较好的信噪比。

5.3.2 有关闪烁光纤阵列模型的模拟结果

对于闪烁光纤阵列的模拟，我们主要是针对几个决定图像质量的参数在高能下的模拟，包括系统输出信噪比(SNRout)、探测量子效率(DQE)、调制传递函数(MTF)等[13]。

输出信噪比 $SNR_{out} = N/\sqrt{N}$，在这里 N 为与探测器发生作用的粒子数目，假设噪声分布为泊松分布。首先模拟分析比较闪烁光纤阵列在高能下，不同强度下的输出信噪比情况。模型设定入射光子的能量区域在 1MeV－20MeV 之间，比较了每根光纤入射 X 光子数目分别为 100，1000，5000 三种情况。图 5.12 显示了在此条件下入射光子能量与输出信噪比之间的关系。从此图可以得出入射能量越大，闪烁光纤阵列的输出信噪比越小；同时，入射光子强度越大，输出信噪比越大。因此要在高能条件下得到高信噪比图像，入射光子强度必须较大。

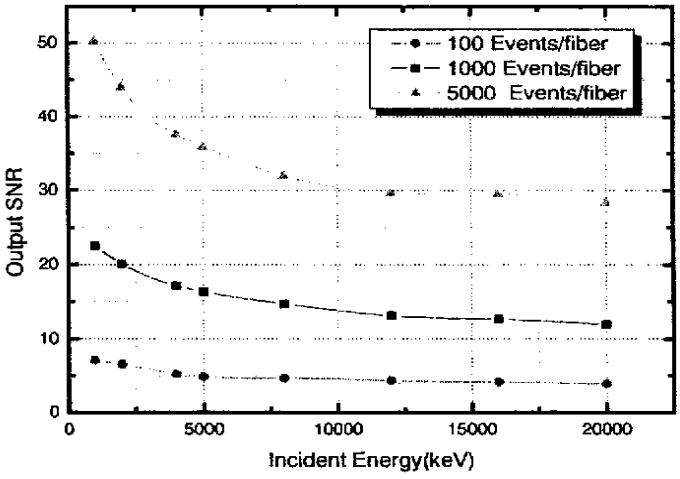


图 5.12 输出信噪比在不同入射强度下与入射能量的关系

我们假设定义的入射粒子数目 N_0 全部击中探测器，则此时的输入信噪比可以简单的定义为 $SNR_{in} = N_0/\sqrt{N_0}$。根据 $DQE(0) = SNR_{out}^2/SNR_{in}^2$，可以模拟计算得到闪烁光纤成像探测器在零频域下，即无被测物体情况下的 DQE 值。

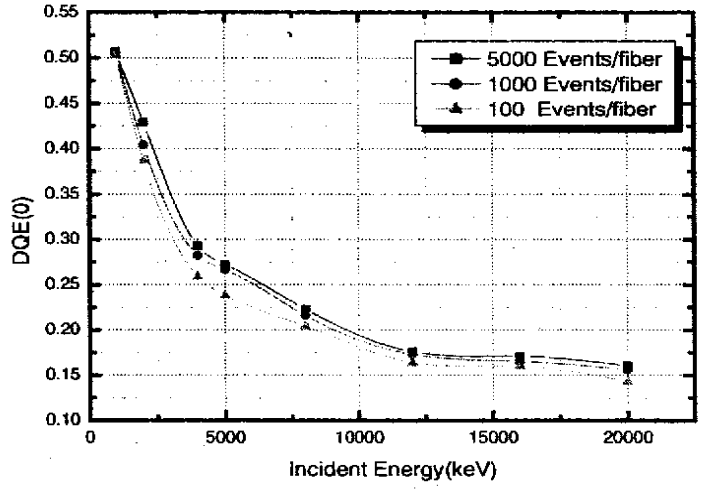


图 5.13 DQE(0)在不同入射强度下与入射能量的关系

图 5.13 为不同入射强度下 DQE(0)与入射能量(1MeV—20MeV)的关系。从图中可以看出，随入射能量的增大，DQE 逐渐减小；入射光子强度越大，DQE 值越大。此结论与输出信噪比一致。

当考虑存在被测物体下的 DQE 值时，我们根据公式

$$DQE(f)=\frac{SNR_{out}^2}{SNR_{in}^2}=\frac{S^2MTF^2(f)}{\Phi NPS(f)}$$

可以得到 DQE 的频域曲线。公式中 NPS(f)为功率噪声谱，我们可以通过模拟计算得到。

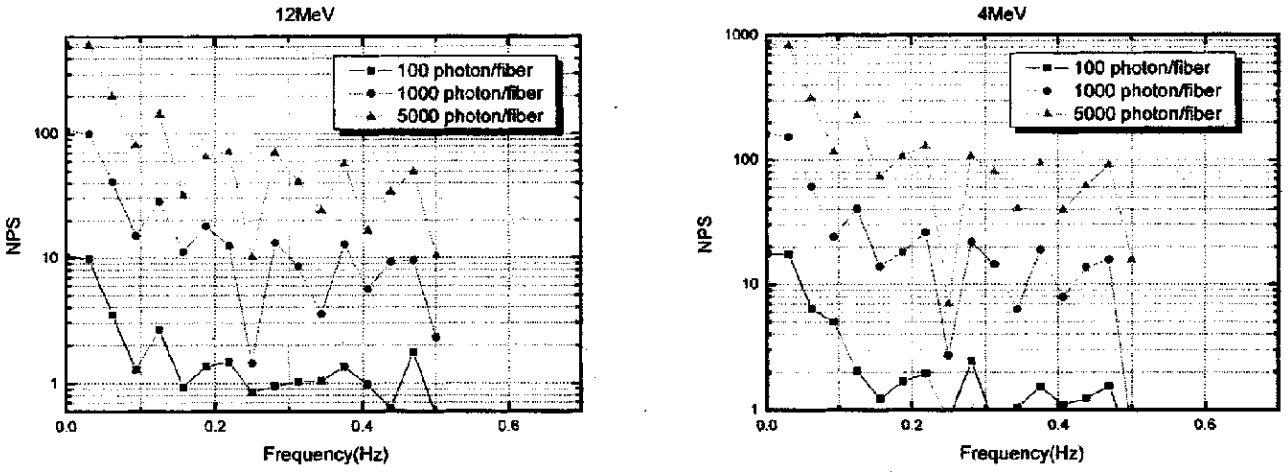


图 5.14 12MeV 和 4MeV 两种入射能量下的 NPS(f)曲线

图 5.14 为入射能量分别是 12MeV 和 4MeV 下的 NPS(f)曲线。通过 NPS 我们可以得到 DQE(f)。

图 5.15 为 12MeV 和 4MeV 入射能量下的 DQE（f）曲线。

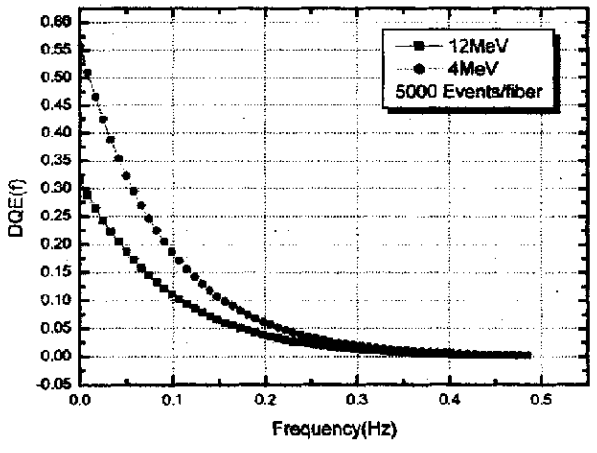


图 5.15 12MeV 和 4MeV 两种入射能量下的 DQE(f)曲线

从图 5.15 看出，4MeV 下的 DQE(f)比 12MeV 大。因此，无论是 DQE(0)，还是 DQE(f)，

都是在低能条件下比高能条件下的值大，强度大条件下比强度小条件下的值大。因此欲在高能时得到较好的图像对比度，必须有高强度的光源。

在本章开始提到传统闪烁体吸收效率的提高与保持较好的调制传递函数(MTF)之间存在矛盾，所以我们模拟了闪烁光纤阵列探测器在高能下的 MTF 情况，以试图寻找新型的成像探测器在图像空间分辨率上的特性。这种特性的研究是闪烁光纤阵列是否能用于高能 X 射线成像的关键。

根据第三章有关 MTF 的计算方法，我们的模拟选择了台阶法测量。首先在建立的模型基础上设定被测物体为一个台阶型，通过模拟直接得到此台阶的边缘扩展函数 ESF，然后根据 ESF 求得 MTF。图 5.16 为直接获取 ESF 的示意图。

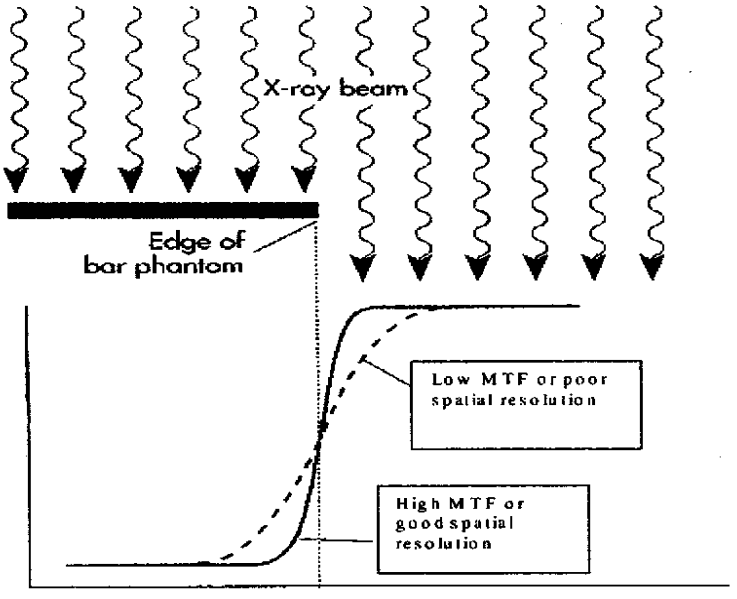


图 5.16 采用台阶法计算 MTF

首先我们模拟了四种不同光纤长度情况下，入射能量为 4MeV 的 MTF。模拟结果如图 5.17 所示。图中横坐标 lp/mm，表示对线/毫米，即每毫米内能分辨的对线个数，这是一种衡量图像分辨率的常用单位，其含义相当于频率。模型中建立的光纤直径 d 为 1mm，因此得到最大分辨率频率，即奈奎斯特频率为 $1/(2d)=0.5$。

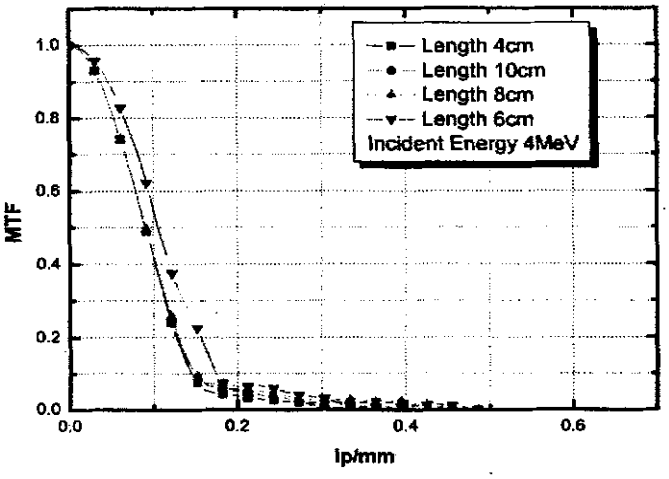


图 5.17 不同长度下的闪烁光纤阵列 MTF 模拟结果

从上图可以看出，不同长度的闪烁光纤阵列，对于某个较高入射能量，所成图像的空间分辨率随光纤长度改变变化不大。这一特性与传统的胶片成像探测器完全不同。这就使得我们可以通过增加闪烁光纤长度来增加探测器吸收效率，同时又不影响图像的空间分辨率。闪烁光纤阵列的这一特性对于高能 X 射线成像具有重大的意义，使得将闪烁光纤阵列成像探测器用于高能成像成为可能。同时，我们还模拟比较了不同的高能入射能量下 MTF 曲线的变化。图 5.18 为分别在 4MeV，10MeV，12MeV，20MeV 四种入射能量下，长度为 10cm 的闪烁光纤阵列的 MTF 曲线。

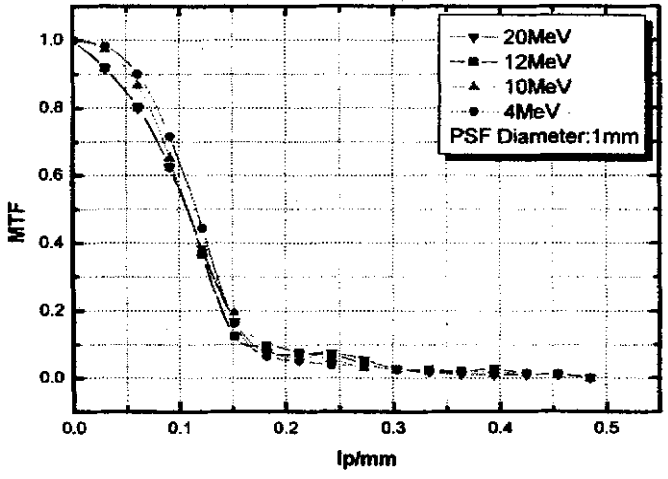
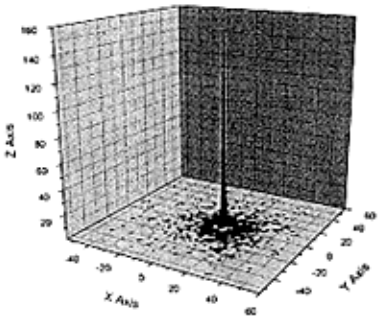


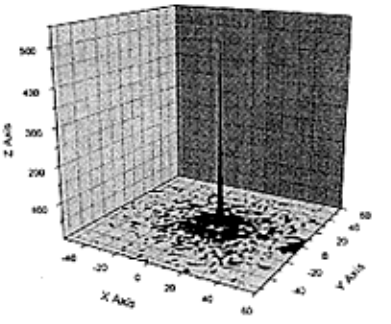
图 5.18 不同入射能量下的闪烁光纤阵列 MTF 模拟结果

从此图可以看出，闪烁光纤阵列的 MTF 基本不受入射能量高低的影响。结合图 5.17 和图 5.18 来看，欲将其运用在高能成像完全可以做到既增加探测器吸收效率，又维持原有的系统空间分辨率，从而解决了困扰高能成像技术的主要矛盾。

在实际闪烁光纤阵列成像探测器中，空间分辨率不是取决于探测器象素单元的大小，即闪烁光纤大小，而是取决于阵列中光纤之间的串扰程度。在高能 X 射线入射下，闪烁光纤的 Clad 包层和 EMA 吸收层不能完全把侧向的散射光子吸收，而是与阵列中邻近的某根闪烁光纤发生作用而产生可见光子，这部分由于侧向散射光子而导致的输出可见光子就是光纤之间的串扰。我们模拟了两种入射能量下，入射光子只对准阵列中某一根光纤，然后计算出阵列中所有闪烁光纤的能量沉积大小，以比较这两个不同入射能量时，阵列中光纤之间的串扰程度大小。



1MeV



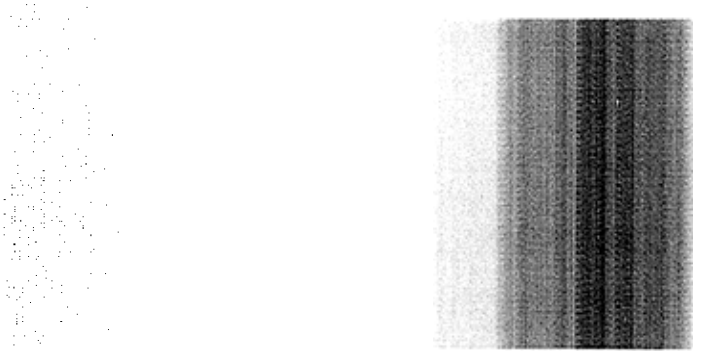
4MeV

图 5.19 1MeV 和 4MeV 入射能量下的串扰程度比较

图 5.19 为入射能量分别为 1MeV 和 4MeV 入射光子对准阵列其中一根光纤时产生的串扰的比较。图中 Z 轴为能量沉积值，XY 轴为光纤阵列的平面位置坐标。图中能量沉积值较大的对应为入射光纤位置以及周围邻近光纤的位置。从图中 XY 平面看到，入射能量较大时，阵列中产生能量沉积的光纤数目较多，即有可见光子输出的光纤数目较多，这样表明阵列的串扰程度较大。因此采用措施抑止或避免串扰对于高能成像具有重要的意义。

串扰极大的影响图像对比度和空间分辨率，我们应该尽量避免串扰的发生。在模拟中，我们试图在结构上对一般闪烁光纤阵列进行一些改造。我们改变了闪烁光纤之间介质，由原来的空气变为铅，以吸收侧向散射光子，从而来检验是否对抑止串扰有所贡献，使得图像质量有所提高。这在对闪烁光纤阵列探测器在高能成像领域是一种新的尝试。

被测物体为如前面所建立的 5 层不同厚度的金属块模型，通过 Geant4 模拟得到原始数据，然后利用一个简单的 DR 图像重建软件得到最后图像。图 5.20 是闪烁光纤之间为空气介质和铅介质的比较。从图中看出在采用铅介质后，图像的对比度和空间分辨率都较好于空气介质时的图像质量。



空气介质　　铅介质

图 5.20 闪烁光纤阵列中为空气介质和铅介质的图像质量对比

从模拟中看出加入铅介质后确实有助于串扰的抑止，对空间分辨率有所贡献，但加入铅介质的厚度将是下一步面临的问题。由于 MTF 不仅受到闪烁光纤之间的散射光子的影响，还受 R 值的影响。R 值的定义为：

$$R = \frac{S_{\text{所有光纤}}}{S_{\text{整个阵列}}} \qquad (5.5)$$

其中 S 为截面面积。R 值越大，对 MTF 越有利。

对于铅介质厚度的定义如图 5.21 所示。

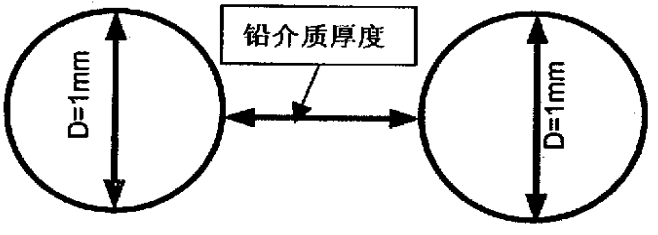


图 5.21 闪烁光纤阵列中铅介质厚度的定义

由式(5.5)得知，当铅介质厚度过大，R 值减小，则对 MTF 产生影响。因此，要达到加入铅介质后既能因减小串扰而提高图像质量，又不能因 R 值的减小而降低图像质量，我们必须在这中间找到一个平衡点。我们选取了几个不同厚度的铅介质来模拟 12MeV 入射下的 MTF，并与空气介质情况下的 MTF 进行比较。图 5.22 为模拟结果。

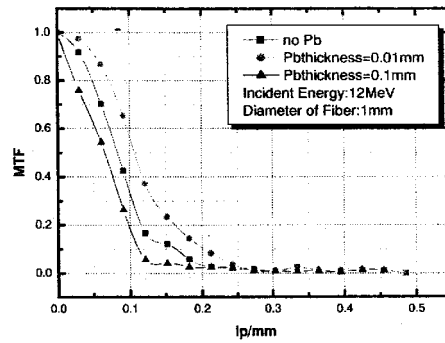


图 5.22 阵列中不同铅介质厚度下的 MTF 曲线

从图中看到，当铅介质厚度为 0.01mm，即光纤直径的 1%时，其空间分辨率 MTF 比空气介质下的 MTF 好；而当铅介质厚度为 0.1mm，即光纤直径的 10%时，空间分辨率较空气介质情况下差。这说明确实在闪烁光纤阵列中加入铅介质后对抑止串扰，提高图像空间分辨率是具有积极的作用的，但同时铅介质的厚度又不宜过大，否则将反而起到消极的作用。

5.4 总结

通过以上对单根闪烁光纤的辐照特性以及闪烁光纤阵列的成像质量的评定和模拟可以看出，在高能 X 射线条件下，虽然由于闪烁光纤材料本身原因，探测效率并不高，但可以通过增加入射光子强度和增加光纤长度等措施来改善这一缺点；同时，由于闪烁光纤在高能下有着更为重要的优点，即光纤长度与入射能量的变化基本上不改变最后图像的空间分辨率，这就使得可以利用光纤这一独特的性质来很好的完成在高能下的成像，从而很好的解决了传统高能射线成像的种种弊端。在利用闪烁光纤的这一优势的同时，还可以对阵列在某些结构上进行一些改进，避免一些由于这种阵列结构带来的类似串扰等问题的影响。通过模拟，我们可以对某些改造方案进行评定，得到一些指导性的参数，从而对下一步的实验工作提供理论指导。

第六章 一些CT和DR成像的模拟及实验

对于闪烁光纤成像探测器的实验研究，我们分为高能和低能的两种类型。高能X射线成像研究需要的射线源系统较为复杂，一般需要采用高压电子直线加速器作为X射线源，因此在系统的研制过程中不可能在短时间内实现闪烁光纤阵列探测器的一系列成像实验。而在较低能的情况下，我们可以采用现有的实验装置以及探测器设计实现一系列的CT和DR成像实验，并结合模拟计算，达到理论与实践相结合的目的，从而对闪烁光纤阵列成像探测器有更深一步的研究和认识。

6.1 CT和DR实验的模拟模型建立

为了简化模拟，我们在建立闪烁光纤阵列模型时采用一维闪烁光纤阵列，由100根直径为0.25mm的光纤水平排列组成，并采用加入铅介质的方法来减少串扰。被测物体的形状和尺寸完全按照实验条件下的真实物体来定义。其形状大致为三个不同直径大小的圆柱体构成，材料为金属铝。每根光纤接收的X光子数为1000。射线源为点源，因此射线束呈扇型排列[12]。图6.1为此一维闪烁光纤阵列以及被测物体的模拟模型图。从图中一维闪烁光纤阵列的结构来看，虽然在成像时扫描次数要比二维阵列要多，但一维阵列的结构决定了其光纤之间的串扰程度要大大小于二维阵列。因此在某些场合采用一维阵列结构不失为一种既简单又实用的成像方法。

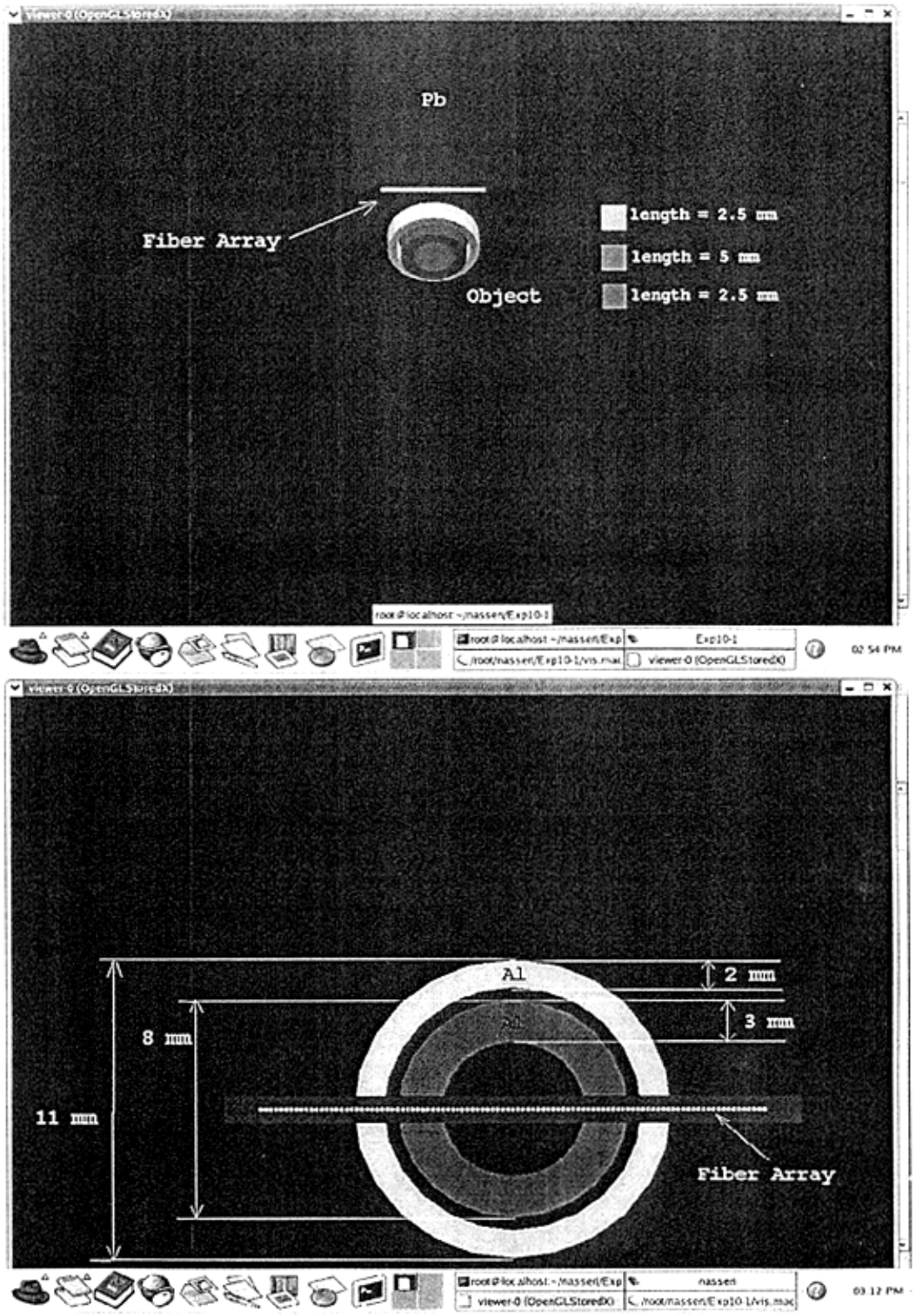


图 6.1 一维闪烁光纤阵列探测器及被测物体的模拟模型图

在模拟中，我们分别采用不同的入射能量和不同的移动步长来对闪烁光纤阵列 CT 和 DR 成像进行研究。对于 CT 成像来说，被测物体在每轮扫描之后，需要进行一个角度的旋转进行下一步扫描，而这个步长越足够小，重建后的图像越清晰。同样，DR 成像每轮扫描之后需要在上下移动一定距离进行下一轮扫描。

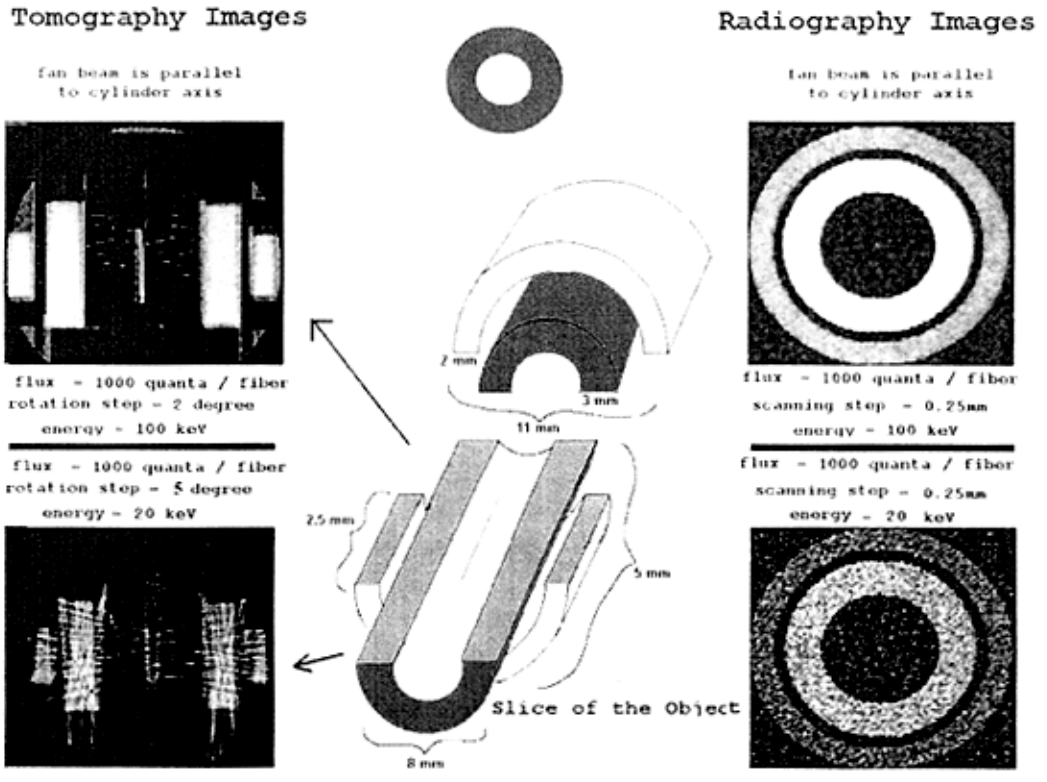


图 6.2 在设定不同条件下模拟得到的物体 CT 和 DR 图

从图 6.2 中显示了不同条件下的 CT 和 DR 图像质量的比较。此图左侧为该物体的 CT 图像。可以看到，左侧上端图像较下端图像更加清晰，其主要原因在于旋转步长更小；而右侧上端图像清晰的主要原因在于入射能量较大，位于被测物体后面的闪烁光纤探测器接收到的穿越粒子数目较多，使得图像信噪比较大，因此图像质量较好。这些图像的重建均是采用前面章节提到的由我们自行研制开发的基于滤波反投影法 CT 重建算法的应用程序。该程序需要的输入数据既可以是模拟数据也可以是实验数据。

在实验时，我们还选取了另一个物体作为被测物体，并也针对它做了模拟。该物体由一个铝材料的圆柱空心体，中心处为一个直径为 0.25mm 的铜棒。从图 6.3 的 CT 和 DR 图可以清楚看到此铜芯棒。

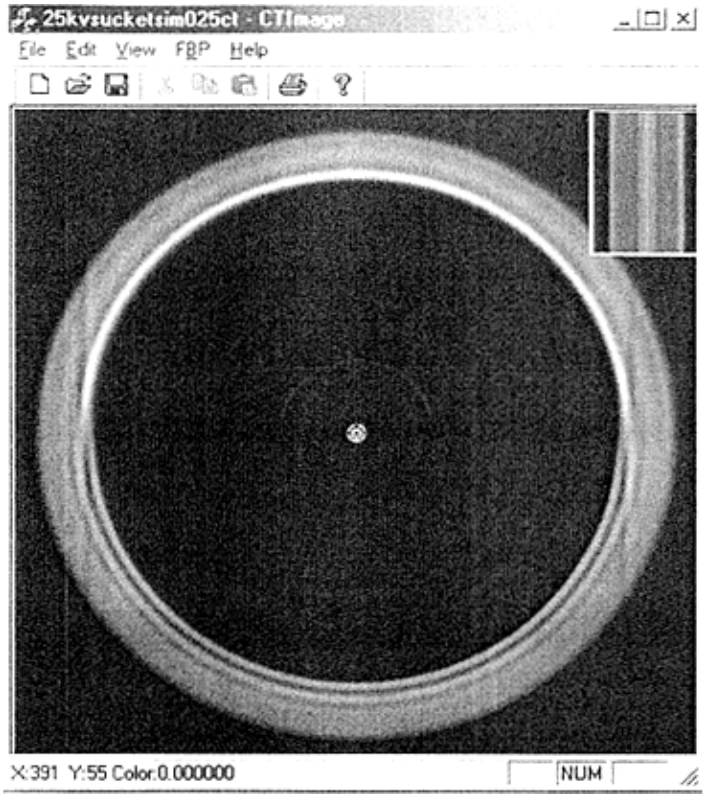


图 6.3 带铜芯棒的铝圆柱体 CT 和 DR 模拟图

在实验装置设计中，一维闪烁光纤阵列的输出端被捆在一起，这样可以较容易的与 CCD 相连。CCD 相机具有自己的软件上或硬件上的帧获取单元，这些单元能够对每个像素数据分别进行存储。示意图如图 6.4 所示。通过图像处理软件，对 CCD 存储的各个像素数据进行分离，最后还原成原始的线性光纤数据。

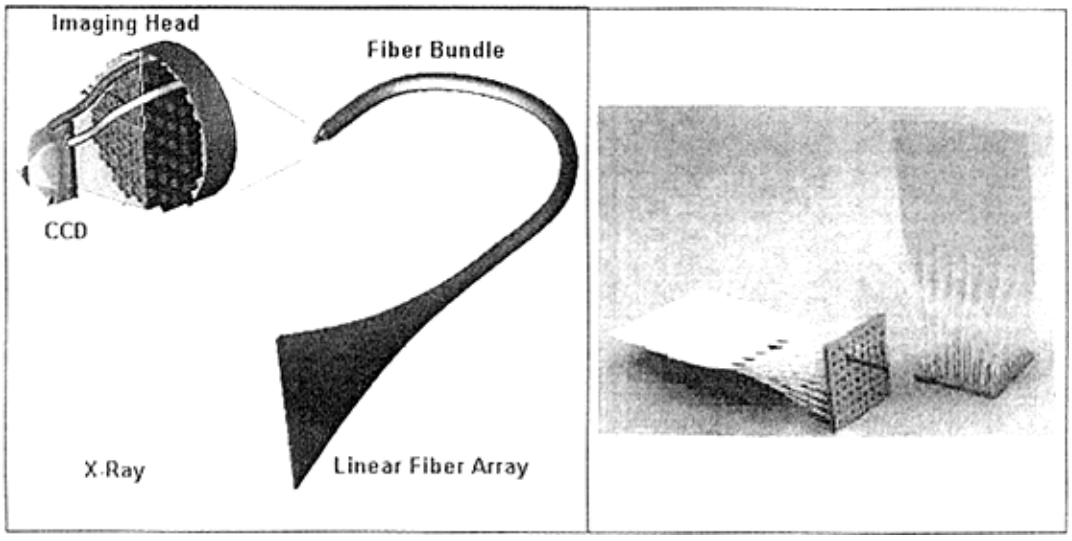


图 6.4 一束线性闪烁光纤以及后续 CCD 读出系统

我们模拟了 CCD 在接收线性闪烁光纤阵列输出的可见光子时的分布情况。处于简便的考虑，我们定义光纤直接与 CCD 相连。模型如图 6.5 所示。

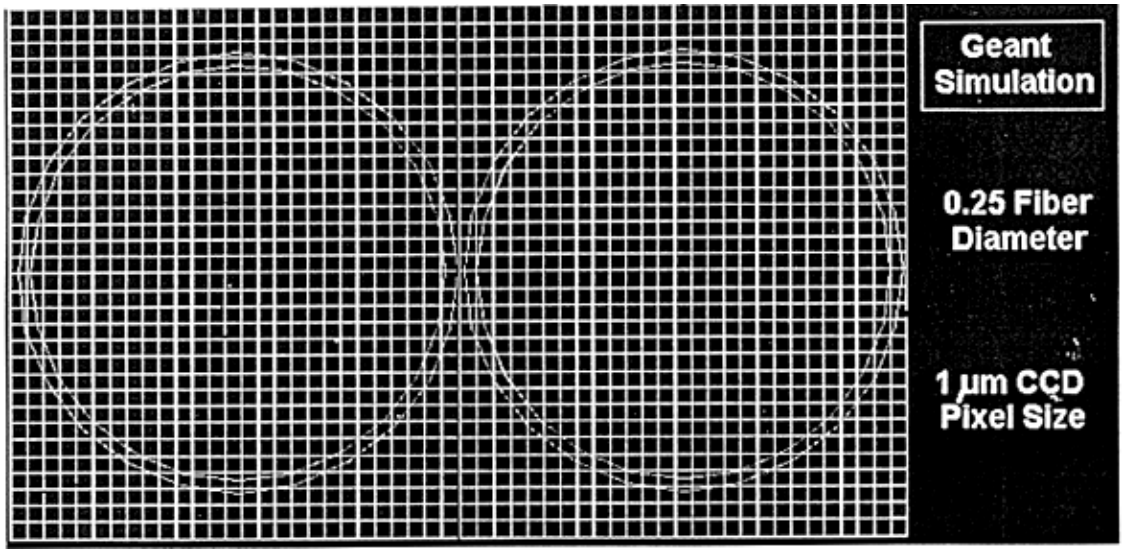


图 6.5 闪烁光纤直接与 CCD 相连测量可见光子在 CCD 像素上的分布的模型截面

从光纤出来的可见光子将被许多 CCD 像素接收到，每个像素接收到的可见光子数目不同。位于光纤正后面的像素区域获得的可见光子较多，周边像素区域获得的较少。图 6.6 为定义的模型以及 CCD 获取的可见光子数目。

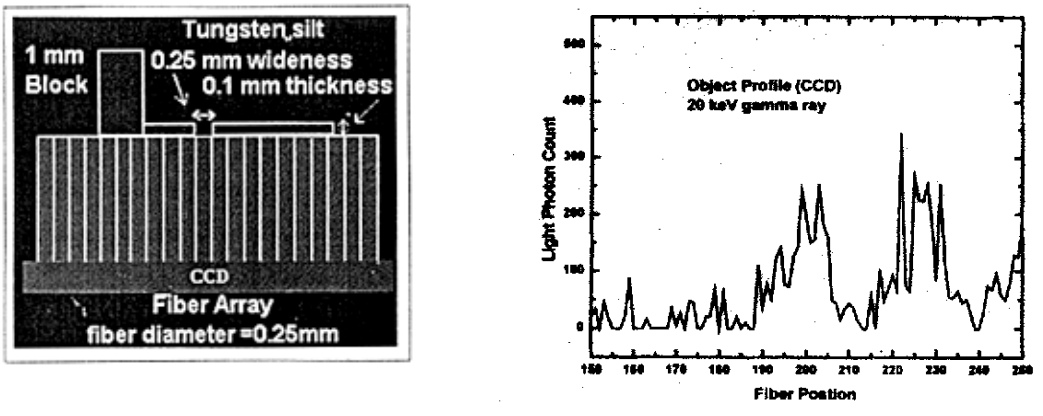


图 6.6 测量 CCD 获取的可见光子数目的模型以及模拟结果

图中可见光子数目包含由于测向散射光子影响而导致的噪声。这种直接对 CCD 像素的可见光子数目分布的模拟方法，其结果比能量沉积模拟方法更加贴近实验真实结果。

6.2 CT 和 DR 成像的实验介绍

所有的射线成像系统都是由以下三部分组成：射线源，探测器系统和读出系统。一般低能 X 射线源由 X 射线发生器或放射性元素产生。探测器系统主要是一个能产生电子或可见光子的探测器，这些次级粒子信息的测量通过读出系统得到。

在我们的实验装置中，探测器系统部分主要是采用一根圆柱形塑料闪烁光纤，一个光电倍增管和示波器作为读出系统[5]。探测器采用的这种方法原理上实际与由光纤阵列组成的方法是相同的。这种单根光纤成像方法的最大优点就是消除了光纤单元之间的串扰以及散射效应。

图 6.7 为基于单根光纤成像的实验装置图。

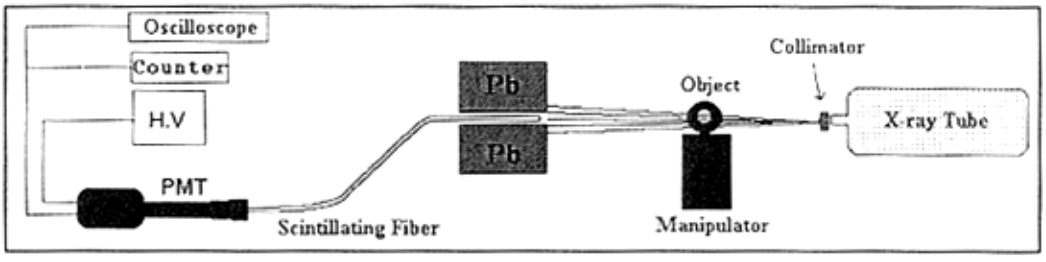


图 6.7 基于单根光纤成像的实验装置示意图

从上图中看出此成像系统是主要由以下结构组成：

1）X 射线发生器

我们采用的是医疗牙科用的 X 射线发生器作为一个简单的 X 射线源。该射线发生器的阳极靶使用的是金属钨，能产生电压峰值 65 kVp，电流 1.5mA。焦点大小：0.3mmX0.3mm。X 射线通过一个直径为 2mm 的小孔进行准直。射线过滤器采用一个厚度为 1.5mm 的铝板，其作用是尽可能完全消除能量低于 20keV 的 X 光子。正是由于过滤器的存在，从发生器出来的 X 射线能谱变窄，从 20keV 到 65keV。在这里由于过滤器使得谱峰位置向高能方向移动而导致的束谱硬化问题被忽略，而该实验选取的被测物体厚度也不大，因此也无需考虑束谱硬化。

2）操作平台

被测物体放在平台上面能够在水平和垂直方向自由的移动。它在系统中位于 X 射线源和闪烁光纤输入端之间。

3）闪烁光纤

实验中我们采用的闪烁光纤为 Bicron 公司的 BCF-10，直径为 1mm，长度选取了 20cm。BCF-10 的其它部分参数如图 6.8 所示。

Specific Properties of Standard Formulations

Fiber	Emission Peak nm	Decay Time ns	1/e Length m (1)	Number of Photons per MeV (2)
BCF-10	432	2.7	> 1.9	~8000
BCF-12	435	3.2	> 2.2	~8000
BCF-20	492	2.7	> 3.5	~8000
BCF-60	530	7	> 3.5	~7100
BCF-91A	494	12	> 3.5	N/A
BCF-92	492	2.7	> 3.5	N/A
BCF-98	N/A	N/A	N/A	N/A

BCF-10

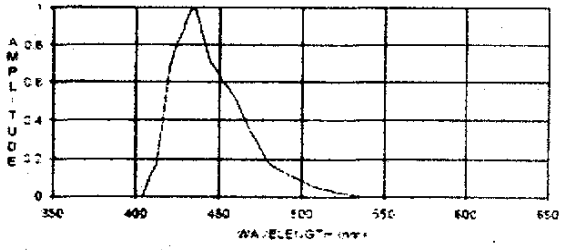


图 6.8 BCF-10 闪烁光纤的参数

4）光电倍增管

实验采用的光电倍增管(PMT)是日本 HAMAMATSU 公司生产的 R1166。对于波长范围在 300—650nm 的辐照，其阴极灵敏度为 105uA/lm。它的波长响应谱峰值为 420nm，而闪烁光纤 BCF-10 的发射谱峰值为 432nm。在这个波长下，光电倍增管的量子效率为 25%，已达到此光电倍增管的最大效率。R1166 光电倍增管的效率曲线如图 6.9 所示。

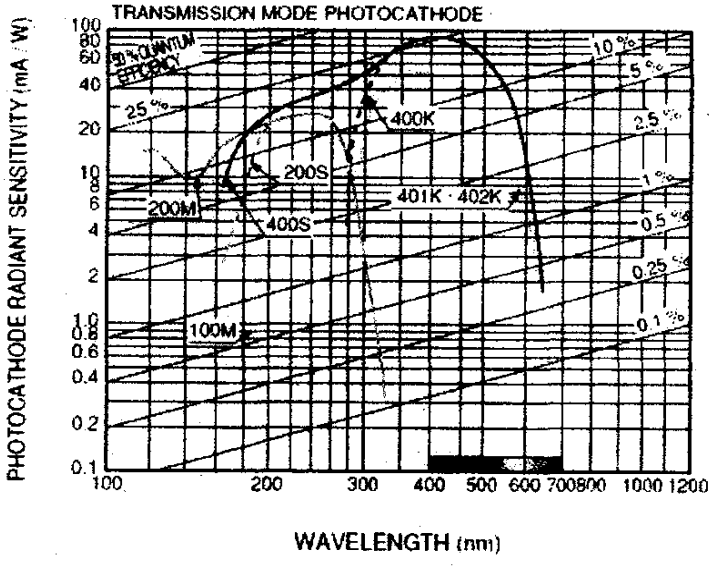


图 6.9 R1166 光电倍增管的效率曲线（曲线 400K）

5）示波器

本实验中采用 Tektronix 的 TDS 3032 示波器，用来高精度的记录光电倍增管输出信号的幅度。

6）高压电源

实验中光电倍增管的高压操作电源为 900－1100V。

图 6.10 是整个基于单根闪烁光纤成像系统的实验装置俯视图。

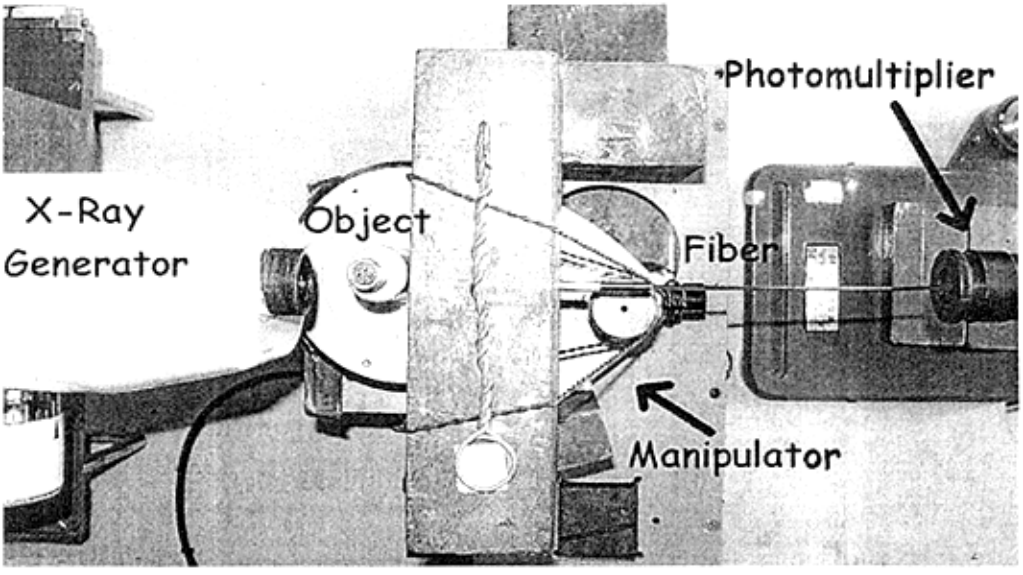


图 6.10 基于单根闪烁光纤成像系统的实验装置图

此成像系统的基本原理如下：X 射线束从射线发生器里出来穿透被测物体后与单根闪烁光纤发生作用，产生能量沉积并使光闪烁光纤处于激发态，在退激发时产生波长处于可见光波段的电磁波。产生的可见光子基于全反射在光纤中传播到输出端，之后进入光电倍增管产生电信号。电信号的大小由射线能量，辐射强度以及闪烁光纤的光学属性等因素决定。实验中，通过射线源的能谱平均值（27－33keV）、闪烁光纤产生的可见光子能量（约 3eV）、闪烁光纤的俘获效率和传输效率以及光电倍增管的量子效率，我们可以估算得到此系统中每一个 X 光子产生 0.6 个可见光子。由于这个数值较小，易受外界影响，因此在实验中必须采取措施防止外界环境中的可见光进入光纤，例如可以考虑用黑色屏蔽物把整个闪烁光纤罩住。否则即使很少量的外界可见光的混入都可能导致很严重的图像噪声，并且对光电倍增管也有一定程度的破坏。

光电倍增管出来的信号若足够大，我们可以通过示波器进行测量和读数；若信号较小，我们可以采用一个放大器来提高信号幅度。图 6.11 为实验中在示波器上显示的某输出信号

的波形。

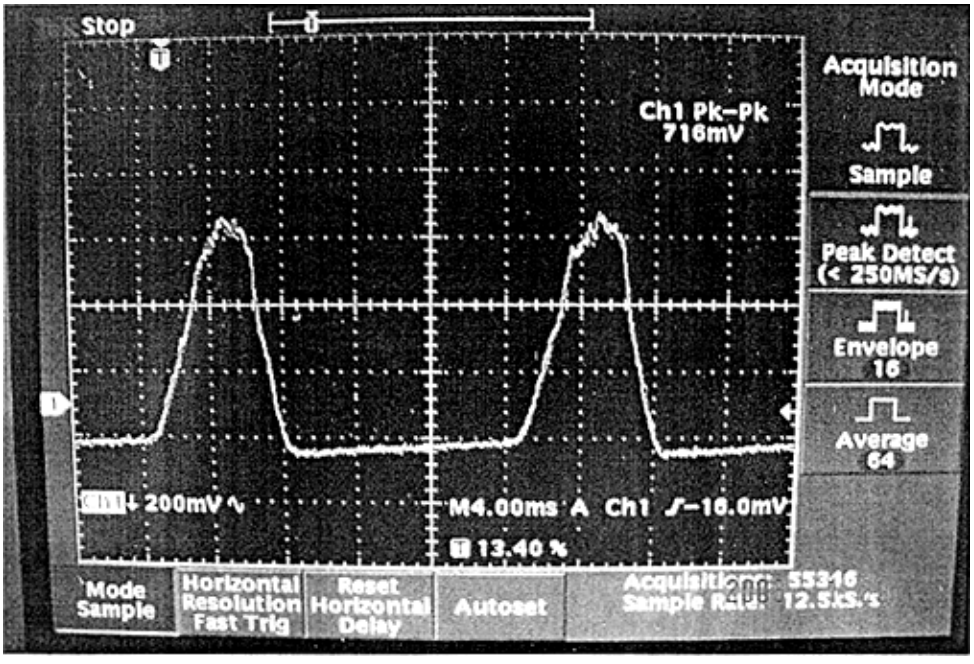


图 6.11 实验中示波器显示的光电倍增管输出信号波形

从图 6.7 和图 6.10 我们看到，这根闪烁光纤的输入端放置在铅块上的一个直径与光纤差不多大小的小孔里，以屏蔽其它散射 X 光子的干扰，同时将射线源窗口对准铅块上的小孔，并且使它们与被测物体处于一条水平线上，以确保光纤的探测效率最大；而光纤输出端与光电倍增管直接相连。为了使闪烁光纤的避免外界可见光的干扰，在实验时，我们用一个黑色的屏蔽胶带把整个光纤包住。

实验中，我们选取了四个被测物体来说明此成像方法的有效性。这个四个物体分别为：样品 A——个铝质的圆环柱体，外环直径 14mm，内环直径 6mm，长度 11mm。在圆环柱体外包有一层 1mm 厚的塑料体；样品 B——个陶瓷的圆环柱体，外环直径和内环直径分别为 9mm 和 5.5mm；样品 C——个带插针的电子接口器件，有铝铜合金制成，外面圆柱体的外环直径和内环直径分别为 5.5mm 和 4.5mm，中心处的针直径 1mm；样品 D——也是一个带插针的电子接口器件，外围是铝质的圆柱体，外环直径和内环直径分别为 11.5mm 和 9mm，中心处是四个铜质的针，其直径为 1.5mm。

在进行被测样品各个方向的扫描时，我们不是采用移动射线源位置或直接移动样品，而是将样品放在一个能在水平垂直方向自由移动的操作平台上，并且移动距离可以根据平台上的刻度来进行。对于 CT 成像过程中，样品水平移动的步长是 1mm。在一次水平扫描完成

成之后，样品需要以垂直中心轴旋转一个很小的角度，一般此角度步长不超过2度，再进行一次水平方向的移动扫描，完成之后再进行小角度的旋转，依此类推，最终完成360度的旋转。对于DR成像，当一次水平线方向扫描完成之后，样品还需要在垂直方向移动一个步长，接着再水平扫描一次，依此类推。实验中，每一步需要的测量时间为2s（X射线发生器暴光时间），然后每一步对样品的移动时间大约为10s，这样每一步花费的总时间为12s。当然，这个时间并不是绝对的，它取决于样品的本身大小以及系统对分辨率的要求。假如采用一个快速的步进机，这个时间将可能减小到约4sec/step。

对于样品A，B，C，D，在射线源后放置的方向如图6.12所示。可以看到样品A圆柱体中心轴方向与射线方向平行，而另外三个样品的圆柱体中心轴方向与射线方向垂直。

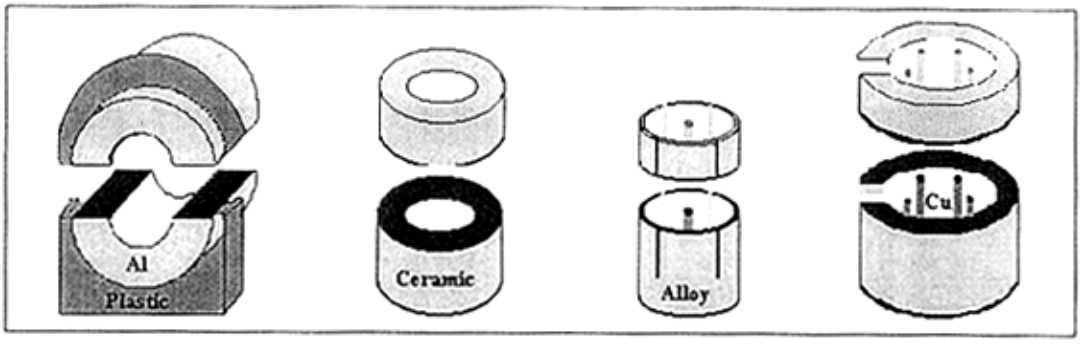


图6.12 四种样品在成像时在射线源后放置的方向示意图

图6.13为四种样品的实物图以及与之对应的CT成像实验结果（注：图中上方的实物图与下方的实验结果图并不是同一个比例）。从图中看到物体所有的细节从下方的CT图像中都能体现出来。对于样品A，假如对比度再大一些，实物图中的外围塑料部分将会看得更加清晰；对于样品B，此物体相对简单并且非常对称，CT表现出来的材料也很统一；对于样品D，圆环上的缺口以及中心的四个针在CT图中都清楚的表现出来。

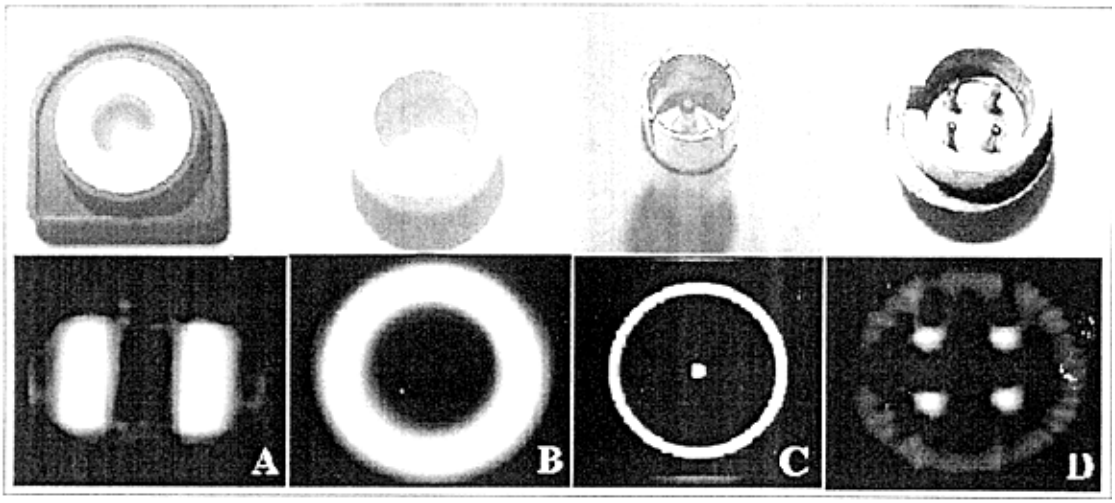


图 6.13 四种样品的实物图以及与之对应的 CT 成像实验结果

在第三章我们提到，成像系统的质量可以通过调制传递函数 MTF 来衡量。而 MTF 是首先通过测量得到系统的边缘扩展函数 ESF，然后对 ESF 进行处理计算获得。图 6.14 为此成像系统实验的 MTF 曲线。

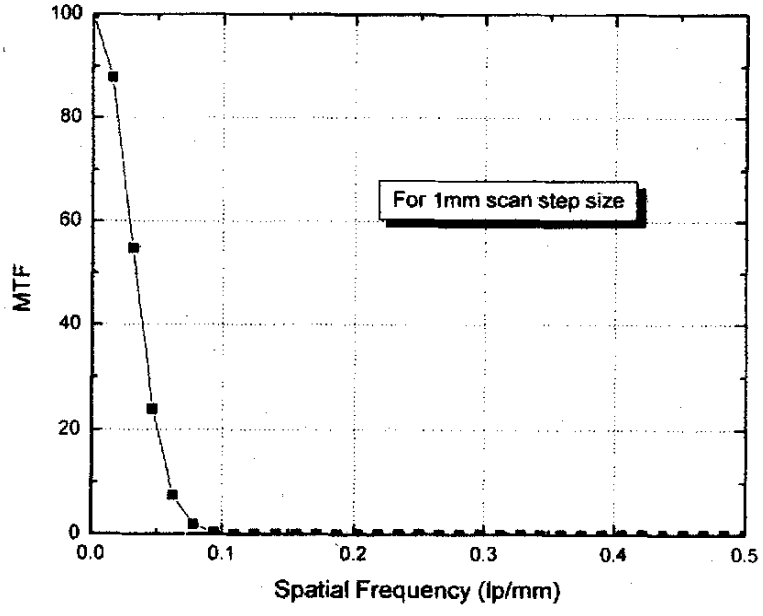


图 6.14 单根闪烁光纤成像系统实验的 MTF 曲线

从此 MTF 曲线图看出，对于被测物体形状变化频率大于 0.1lp/mm 时，系统 MTF 几乎达到零，即系统不能分辨此频率变化；而在被测物体形状变化频率为 0.05lp/mm 时，系统

MTF 约为 20%。总体来说，对于单根闪烁光纤成像系统，扫描步长和光纤直径越小，得到图像的空间分辨率越好。

实验中我们还测量了系统的信噪比与被测物体厚度的关系。我们对每个厚度的铝块进行多次测试，每次在示波器上读出光电倍增管出来的信号，然后对多次得到的信号求平均值。而对于噪声，普通的求法是假设噪声分布符合标准泊松分布的前提下对有效信号求均方根。得到有效信号和噪声大小后可以得到系统的信噪比。图 6.15 为系统的信噪比与被测铝块厚度的关系。

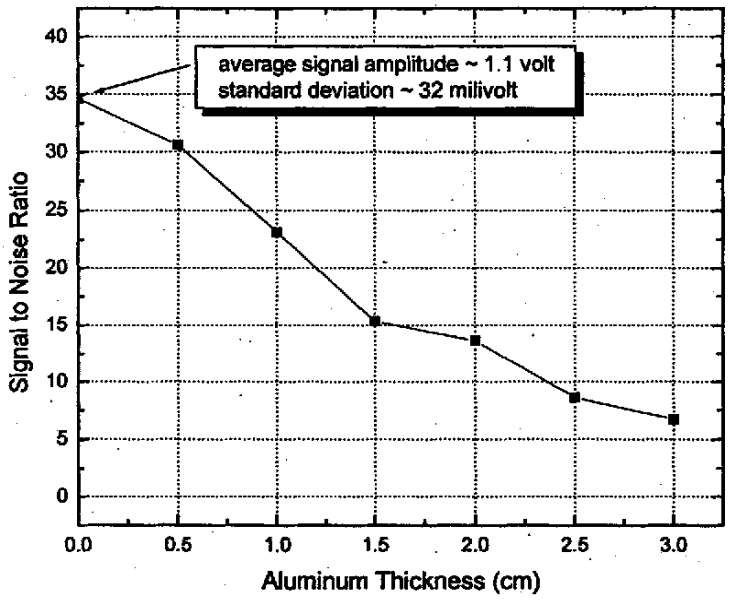


图 6.15 系统的信噪比与被测铝块厚度的关系

从图中看到当铝块厚度为 3cm 时，此成像系统的信噪比值只有 5 左右。因此，要想提高成像系统的信噪比，一般可以采用以下两种方法：使用更高强度的 X 射线发生器；减小被测物体的厚度。

以上主要介绍了基于单根闪烁光纤的射线成像系统。这种系统的主要特点是装置简单，原理清晰，非常适合科研单位进行小范围成像实验研究时采用。它得到的最后图像空间分辨率取决于所采用光纤本身的大小以及被测物体移动的步长。通过这个实验也能对闪烁光纤作为成像探测器有一个更全面深入的认识。

针对此成像系统，我们主要进行了有关 CT 的成像实验。关于 DR 的成像实验我们在第二章中基于二维闪烁光纤阵列的成像实验中有较多的介绍，在这里就不赘述。

第七章 总结与展望

闪烁光纤阵列探测器在射线成像领域开辟了一种全新思路，尤其在高能成像领域。本论文综合了闪烁光纤探测技术、CCD 光电转换技术、图像重建技术和计算机模拟技术，给出了在较大能量范围内对被测物体进行实时的CT或DR成像，并对闪烁光纤在这些能量范围内的辐照特性和传输特性做了深入的理论研究，为闪烁光纤阵列成像探测器应用于医疗放射成像，工业瞬态成像以及工业无损探伤等提供了非常有价值的研究和开发方法。

通过对基于闪烁光纤的成像探测器从理论到实验的研究，我们获得了许多具有重要价值的结论：

- 闪烁光纤长度与入射能量的变化基本上不改变最后图像的空间分辨率，这就使得可以利用光纤这一独特的性质来很好的完成在高能下的成像，从而很好的解决了传统高能射线成像的种种弊端。
- 在高能入射条件下，采用闪烁光纤阵列成像探测器时，射线源强度越大，系统信噪比和量子探测效率越好，所得到的图像质量也越好。
- 闪烁光纤在吸收效率上存在长度饱和度。闪烁光纤吸收效率可以通过增加光纤长度来改善，但又并不能一味增加光纤长度，这是由于其中存在一个光纤长度饱和度的问题。即当光纤长度增加到10cm左右时，吸收效率几乎不再增加，达到饱和状态。
- 理论分析得到在闪烁光纤阵列中加入适当厚度的铅介质的方法对较少光纤之间的串扰，提高图像质量具有积极的作用。并且总结出实际成像探测器系统的空间分辨率并不是取决于光纤的大小，而是取决于阵列中光纤之间散射粒子的串扰程度。
- 采用单根闪烁光纤实现了CT和DR成像。并且得到扫描步长，被测物体厚度，以及入射粒子能量对图像质量都有不同程度的影响。

另外，通过对闪烁光纤阵列探测器在高能入射粒子下的计算机模拟，我们获得到了一组闪烁光纤阵列成像探测器在高能下的特征参数值。如表7.1所示。

表 7.1 较高能下闪烁光纤成像探测器的特征参数模拟值

能量范围	1-4 MeV	12-20MeV
量子效率	~40%	17%
优化长度	~8-10cm	10-12cm
灵敏度（或 DE） (light photon / keV)	~0.06	0.008
MTF* (20%)	0.15 lp/mm	0.13 lp/mm
DQE (20%)	0.1lp/mm	0.06lp/mm

***Limiting Resolution ~0.31lp/mm or 1.6mm**

***BCF-20, Diameter=1mm**

在全面总结的同时，我们对该成像方法在下一步的研究设想和未来的前景提出如下的思考和建议：

- 闪烁光纤的进一步研究

作为探测器核心的闪烁光纤阵列，其对探测系统的整体性能起着决定性的作用。对闪烁光纤进行进一步深入的研究是十分必要的，特别是对闪烁光纤的材料、几何形状、几何尺寸、线性范围、辐照灵敏度等的研究尤其重要。

- 充分利用计算机模拟技术

在现代测量技术的研究工作中，计算机模拟已经成为一种重要的手段，充分地利用计算机模拟技术，可以帮助我们有效地分析和解决问题。因此下一步可以继续结合计算机模拟技术对闪烁光纤各种特性展开研究。

- 探测器效率问题

从已有的研究可以看出闪烁光纤在高能下存在探测效率和吸收效率的不足，虽然可以通过增加光纤长度或提高射线源强度来加以改善，但在某些情况下效果并不明显，因此如何提高效率成为下一步的研究重点。初步预想通过选择一些加入重金属和高发光效率的无机闪烁体的闪烁光纤来考察。

- 散射光子的串扰问题

在高能成像条件下，解决散射光子的串扰问题极为重要，直接决定图像的空间分辨率。研究中，我们采用的在阵列中加入铅的方法需要专用高精度设备，且实现起来较为麻烦。因此找到一种相对较为简单的解决串扰的方法成为下一步工作的另一个重点。

- 应用于中子成像领域的设想

由于闪烁光纤的主要成分是氢和氧，而经过一些模拟研究得知这两种元素对于中子的吸收和探测效率非常高，远大于 X 或γ射线。因此我们可以考虑把闪烁光纤应用于中子成像领域，其潜力巨大。

参考文献

[1] 唐孝威 主编，*粒子物理实验方法*，人民教育出版社，1982 年 2 月

[2] Avinash C. Kak, Malcolm Slaney. *Principles of Computerized Tomographic Imaging* 1999 IEEE Press

[3] B Jahne. *Digital Image Processing*, 《Algorithms and Scientific Application》1991

[4] *Geant User's Guide (4.2)*, CERN

[5] M.M.Nasseri, Zejie Yin et.al. *X-ray imaging using a single plastic scintillating fiber* 《Nuclear Instruments and Methods in Physics Research B》225 (2004) 617－622

[6] 唐瑜 吴孝义，*GEANT3 系统的应用*，《核电子学与探测技术》Vol.20 No.2 2000 p150-153

[7] 唐瑜 吴孝义 阴泽杰，*运用 GEANT3 模拟分析闪烁光纤辐照特性与光纤直径及入射角之关系*，《核技术》Vol.23 No.2 2000 p:105-109

[8] 马文淦 编著，*计算物理学*，中国科学技术大学出版社，2004 年 9 月

[9] 马庆力 吴孝义 阴泽杰，*闪烁光纤在 r 射线辐照下部分特性的蒙特卡罗模拟*，《强激光与粒子束》 Vol.16 No.3 2004 p:385-388

[10] 马庆力 吴孝义 阴泽杰，*运用 GEANT4 模拟闪烁光纤阵列特性与光纤长度及入射能量的关系*，《核技术》 Vol.28 No.2 2005

[11] M M . Nasseri, Ma Qingli et.al. *Monte-Carlo Simulation for Determining SNR and DQE of Linear Array Plastic Scintillating Fiber*, 《Nuclear Science & Technique》 , Vol.15 No.5 Oct.2004.p:304-307

[12] M M . Nasseri, Ma Qingli et.al. *Monte-Carlo Simulation to Determine Detector Efficiency of Plastic Scintillating Fiber*, 《Nuclear Science & Technique》, Vol.15 No.5 Oct.2004. p:308-311

[13] M M . Nasseri, Ma Qingli et.al. *Image quality evaluation of linear plastic scintillating fiber array detector for X-ray imaging*, 《Nuclear Science & Technique》 Vol.15 No.6 Oct.2004 p:361-364

[14] M.M.Nasseri, Qingli Ma, Zejie Yin et.al. *Low Energy X-ray Imaging Using Plastic Scintillating Fiber: a Simulation Study* 《Nuclear Instruments and Methods in Physics Research B》(已接收)

[15] 吴世法.编著 《近代成像技术与图像处理》国防工业出版社,1997.35－43.

[16] H. Shao, D.W.Miller, et al. *Scintillating fiber optics for X-ray radiation imaging* 《Nuclear Instruments and Methods in Physics Research A》, 1990,A299:528-533.

[17] T.O.White, *Scintillating Fibers* 《Nuclear Instruments and Methods in Physics Research》 A273, pp820-825, (1988).

[18] A. Ikhlef, M. Skowronek, *Some Emission Characteristics of Scintillating Fibers for Low Energy X and γ Rays* 《IEEE Trans. On Nuclear Science》, Vol. 41, No. 2, 1994.

[19] S. Costa, P. Ottonello, G.A. Rottigni, G. Zanella, R. Zannoni, *DQE measurement in a scintillating glass optical fiber detector for X-ray imaging* 《Nuclear Instruments and Methods in Physics Research》, Vol. A380, pp.568-571, 1996.

[20] Abdelaziz Ikhlef and Maurice Skowronek, *Application of a plastic scintillating fiber array for low-energy X-ray imaging* 《App. Optics.》, Vol. 37, No. 34, 1998.

[21] E.P.Q.Alcon, R.T.Lopes, *Slot scintillation detector modeling for digital radiography*, 《Radiation Physics and Chemistry》61(2001) 411-414

[22] Chakarova R, *Monte Carlo study of light transport in scintillating fiber*, 《Nuclear Instrument & Methods in Physics Research A》, 1995, 364:90-94.

[23] Stavina P, Tokar S, Budagov A, et al. *Simulation studies of the electromagnetic energy resolution of scintillating fiber calorimeter*, 《Nuclear Instrument & Methods in Physics Research A》, 1995, 364:124-132

[24] 唐淳，杨成龙，江孝国，等，*发光玻璃转换屏空间分辨率研究*，《强激光与粒子束》2000,12(6)：673－676

[25] 谭肇，李泽仁，江孝国。*辐射照相系统中转换屏的分辨率研究*，《强激光与粒子束》，2003，15(3)：302－304

[26] 王义，杜宏亮等。*X射线增感屏的图像信息传输特性分析*，《光子学报》，Vol.28, No.2, 1999 p:155－160

[27] 汪晓莲，许咨宗，汪兆民等。*β射线光纤成像系统*，《核技术》，Vol.20 No.10 1997 p:583－586

[28] 汪晓莲，许咨宗等。*闪烁光纤光传输特性的研究*，《高能物理与核物理》，Vol.21, No.9 1997 p:787－792

[29] 王婉丽，江孝国，吴廷烈等。*台阶法测量CCD成像系统MTF的数据处理方法*，《光电子.激光》，Vol.13, No,2 2002 p:173－175

[30] 王云秀，王婉丽。*台阶法测量CCD成像系统MTF的原理及方法*，《光电子技术》，Vol.22 No.3 2002 p:171－174

[31] A. Ikhlef, M. Skowronek, A.S. Beddar. *X-ray imaging and detection using plastic scintillating fibers*, 《Nuclear Instrument & Methods in Physics Research A》442 (2000) 428-432

附 录

A. 关于 GEANT 系统

1. 简介

Geant 是在 70 年代由 CERN (欧洲核子研究中心)开发的一个软件包.在 3.0 版本以前,Geant 都是属于非公有软件, 只有与 CERN 有科研合作关系的学院或研究部门才有可能获准使用该软件。新一代的 GEANT4 全面采用面向对象的方法设计开发,并已于 1998 年 9 月推出了 Beta2 测试版本。可以从 CERN 网站上下载。目前我们用的是 GEANT4.7.0。

Geant4 是基于蒙特卡罗方法的一个包容大量探测器描述和模拟工具的软件系统。主要应用在三个方面:

1. 对穿越试验装置的粒子进行跟踪
2. 模拟粒子与探测器的相互作用
3. 给出实验装置的几何描述和粒子径迹的图形描述

当时开发后主要用于高能物理,除此之外,现在广泛的运用于核物理,核医学,辐射防护,航空天文等领域。

由于编程方法的改变是的开发方不仅仅局限在 CERN,现在的 GEANT 实际上是来自各个国家开发的库,用户接口组合而成的。因此也延伸了它的功能, 可以实现优化,实验数据分析,重建等功能。

2. 操作系统

1) UNIX (SUN,HP,DEC)
2) Linux with g++,gcc compiler
3) Windows NT/9X VC++ ,Cygwin32(GNU,模拟 UNIX 系统)

至少需要的软件包:

1) C++编译器(g++,gcc,VC++)
2) CLHEP (Class Library High Energy Physics)
3) GNU make (一般系统自带的)和 shell

若要实现图形化显示, 则还需要图形用户接口 (GUI):

1) X-windows;OPENGL;DAWN;
2) MOMO(GGE,GPE,GAG)(用 java 编写的,下载后是.jar 文件,实质就是扩展明为.class 的压缩文件,但在 linux 下不必解压,直接指明 CLASSPATH=……/.jar 即可。要运行此类文件必须先安装 jdk(java 开发软件包),然后运行 java momo 即可。)

需要的分析软件:

1) AIDA;(库文件)
2) JAS(Java Anaysis Studio)

3. 安装说明

首先从网上下载 Geant4.4.0.tar.gz 文件和 CLHEP(Class Library High Energy Physics).tar.gz,

CLHEP 解压完进行编译后,注意库名问题 libclhep-1.7.0.a 要改名,把版本号去掉。

Geant 进行编译以前要进行环境变量的设定.环境变量对于 GEANT 来说很重要。例如:
G4VIS_USE_OPENGLX=1; G4VIS_BULD_OPENGLX_DRIVER=1

需要设置的环境变量:

G4SYSTEM: 所用的系统,如G4SYSTEM=Linux-g++

G4INSTALL: Geant4软件包安装地路径(ex. $HOME/Geant4)

CLHEP_BASE_DIR: CLHEP安装地路径

以上是必须的设置。

可选的设置:

G4LIB: 内核库文件应该安装的路径(default in $G4INSTALL/lib)

G4TMP: 暂时文件(对象文件, 依赖文件)放置的路径(default in $G4WORKDIR/tmp)

G4BIN: 最后可执行文件放置的路径(default in $G4WORKDIR/bin).

G4INCLUDE: 源代码头文件被镜像到的位置路径(default in $G4INSTALL/include)

所有变量可设置在.bashrc文件里.,也可以运行自己的变量文件,但不如.bashrc方便。

设置完毕后可以正式解压,解压后都是..cc 和.hh的源文件。可以开始编译(时间由所用机型决定),编译完后产生大多数源文件的依赖文件(.o)。

4. 源代码介绍

在一个Geant4程序中,基本结构与一般c++程序相似,包括一个主程序main,一个include文件夹,存放所有头文件(类的声明),扩展名为.hh;一个src文件夹,存放所有定义的类,扩展名为.cc.另外还有一些宏文件(.mac)。

1) 如何定义main()

主程序main的基本内容:主要是一些所定义类的初始化。主要包括探测器几何形状的初始化;事件中粒子定义的初始化;事件中进行的物理过程的初始化。所有指向这些类的指针都存储在G4RunManger这个类当中。G4RunManger关心的是所有Geant软件包提供的核心函数,它是是Geant中唯一一个核心管理类,因而必须在main中明确的建立起来,在Geant的树状结构中它处于根的位置,它的作用:

1. 控制主程序流
2. 建立Geant的主要管理类
3. 管理初始化进程
4. 结束Geant主要管理类进程.

第二条和最后一条在G4RunManger的构造函数和析构函数中分别自动执行;第三条通过调用initialize() 方法来执行,第四条通过调用BeamOn函数来执行。BeanOn()后跟的参数是event的次数。我们知道进行一次模拟是一个Run,一个Run包括不止一个event。在Run与Run之间可以改变探测器几何形状,物理过程等参数,而在一个Run运行之间不能改变这些参数。

在Geant4中有两种类型的用户定义类: **user initialization classes and user action classes.** 前者一般用来对应用程序的初始化.而后者一般用于运行过程中. **user initialization classes** 通过调用SetUserInitialization()来指定给G4RunManager,

而**user action classes**通过调用SetUserAction()方法来进行设置.
如下是一个简单的main():

```
#include "G4RunManager.hh"
#include "G4UImanager.hh"

#include "ExN01DetectorConstruction.hh"
#include "ExN01PhysicsList.hh"
#include "ExN01PrimaryGeneratorAction.hh"

int main()
{
  // construct the default run manager
  G4RunManager* runManager = new G4RunManager;

  // set mandatory initialization classes
  runManager->SetUserInitialization(new ExN01DetectorConstruction);
  runManager->SetUserInitialization(new ExN01PhysicsList);

  // set mandatory user action class
  runManager->SetUserAction(new ExN01PrimaryGeneratorAction);

  // initialize G4 kernel
  runManager->initialize();

  // get the pointer to the UI manager and set verbosities
  G4UImanager* UI = G4UImanager::getUIpointer();
  UI->applyCommand("/run/verbose 1");
  UI->applyCommand("/event/verbose 1");
  UI->applyCommand("/tracking/verbose 1");

  // start a run
  int numberOfEvent = 3;
  runManager->beamOn(numberOfEvent);

  // job termination
  delete runManager;
  return 0;
}
```

Source listing 2.1.1
The simplest example of main().

在Geant4程序中有三个类要求用户必须定义的,叫做强制类(mandatory user classes).其中两个是user initialzation classes,另一个是user action classes.并且这两个基类都是抽象的,必须继承这两个基类,获得自己的类.

Mandatory user classes

G4VuserDetectorConstruction

一个完整的探测器的建立应该在这个类中描述。其中建立时应该定义:

1. 探测器所用的材料
2. 探测器的几何形状
3. 探测器灵敏区域的定义
4. 读出模块

G4VuserPhysicsList

在这个类中，主要是定义程序中将要用到的所有粒子和物理过程。另外还有阈值参数

cutoff的设定。

G4VuserPrimaryGeneratorAction

在这个类中，主要任务是创建主事例。这个类有一个公共虚拟函数：generatePrimaries()。这个函数将在每次事例开始时调用。注意，在Geant4中不提供默认事例行为。

Optional user action classes

Geant4中提供了五个user classes：

G4UserRunAction

G4UserEventAction

G4UserStackingAction

G4UserTrackingAction

G4UserSteppingAction

可以根据应用程序的需要，在这些类中的重要位置定义自己的过程或函数。

G4cout and G4cerr

G4cout和G4cerr是Geant4中定义的iostream对象，其用法与普通的cout和cerr相似。因此，输出串可以显示在另一个视窗或存储在一个文件里。

2）如何定义一个探测器几何形状

基本概念：

在 Geant4 中，一个探测器几何形状是由许多的 volumes 组成。最大的 volume 被称为 World volume。它必须在几何形状上包括所有其它的 volumes。每个 volume 都是定义放入之前定义的 volume 中，包括最先定义的 World volume。

每个 volume 是通过定义描述它的几何形状和物理性质来创建，然后把它放入一个范围更大的 containing volume 中。当一个 volume 被置入另一个 volume 中，我们称前者的 volume 为 daughter volume，后者的 volume 为 mother volume。定义 Daughter volume 位置的坐标系取决于 mother volume 采用的坐标系。

为了描述一个 volume 的形状，我们引入了一个 solid 的概念。一个 solid 代表了一个具有形状的几何体，用来定义 volume 几何体的尺寸。例如定义一个立方体的各边长度 10cm，一个圆柱体的半径 30cm，长度 75cm。

为了描述一个 volume 的全部特性，我们采用 logical volume 这个概念。它包括描述它 solid 的几何特性，加上其自身的物理性质：材料；是否包含灵敏探测器单元，磁场等。

为了描述 volume 的位置，我们又引入了 physical volume。用它的定义把一个 logical volume 的拷贝放入一个更大的 containing volume 中。

换句话说，我们要创建一个 volume，我们至少需要做两件事：1）创建 solid。2）创建 logical volume，这个过程需要利用之前创建的 solid，以及加入其它一些属性。

选择一个 solid

为了创建一个简单的box，你只需要定义它的名字和它的三维尺寸。具体例子可以参照源代码中Novice Example N01。在此例子中关于探测器描述的程序段ExN01DetectorConstruction.cc中，你将找到有关box的定义。

```
G4double expHall_x = 3.0*m;
G4double expHall_y = 1.0*m;
G4double expHall_z = 1.0*m;

G4Box* experimentalHall_box
   = new G4Box("expHall_box",expHall_x,expHall_y,expHall_z);
```

Source listing 2.2.1
Creating a box.

这个box的名称为“expHall_box”，X, Y, Z三方向的半高宽分别为3m, 1m, 1m。同样，定义一个圆柱体也非常简单，我们引入G4Tubs这个类。

```
G4double innerRadiusOfTheTube = 0.*cm;
G4double outerRadiusOfTheTube = 60.*cm;
G4double hightOfTheTube = 50.*cm;
G4double startAngleOfTheTube = 0.*deg;
G4double spanningAngleOfTheTube = 360.*deg;

G4Tubs* tracker_tube
  = new G4Tubs("tracker_tube",
               innerRadiusOfTheTube,
               outerRadiusOfTheTube,
               hightOfTheTube,
               startAngleOfTheTube,
               spanningAngleOfTheTube);
```

Source listing 2.2.2
Creating a cylinder.

这个圆柱体的名称为“tracker_tube”，半径为60cm，长度为50cm。

创建一个logical volume：

为了创建一个logical volume，你必须首先得到solid和volume的材料。因此，利用之前创建的box，你可以创建一个简单的充满氩气的logical volume，可以定义为：

```
G4LogicalVolume* experimentalHall_log
= new G4LogicalVolume(experimentalHall_box,Ar,"expHall_log");
```

这个logical volume名字为“expHall_log”。

同样，我们创建一个铝材料的圆柱体，可以这样定义：

```
G4LogicalVolume* tracker_log
= new G4LogicalVolume(tracker_tube,Al,"tracker_log");
```

圆柱体名字为“tracker_log”。

Volume的定位：

如何定位volume？你需要首先得到logical volume，然后必须决定这个volume被置于哪个已存在的volume。之后你要知道这个已存在的volume的中心处在哪里，如何旋转要创

建的 volume。一只明确这些因素，你就可以创建一个 physical volume 了，从而可以得到这个 volume 的位置以及其它一些属性。

创建一个 physical volume：

创建一个 physical volume 是基于已建立好的 logical volume 基础上。Physical volume 是对 logical volume 的定位，这种定位必须满足把 logical volume 放入一个 mother logical volume 中。例如下面这个例子。

```
G4double trackerPos_x = -1.0*meter;
G4double trackerPos_y =  0.0*meter;
G4double trackerPos_z =  0.0*meter;

G4VPhysicalVolume* tracker_phys
  = new G4PVPlacement(0,                          // no rotation
                      G4ThreeVector(trackerPos_x,trackerPos_y,trackerPos_z),
                                                  // translation position
                      tracker_log,                // its logical volume
                      "tracker",                  // its name
                      experimentalHall_log,       // its mother (logical) volume
                      false,                      // no boolean operations
                      0);                         // its copy number
```

Source listing 2.2.3
A simple physical volume.

可以看到 logical volume “tracker_log”被放入到 mother logical volume “experimentHall_log”中，没有旋转。

在定位于 mother volume 这个概念中，存在一个特例。这个就是对于 world volume，这个最大的 volume 来说，由于它包含了所有其他的 volume，而且最早创建，因此它不能被其他 volume 包含，因而它不存在 mother volume，即在创建 G4PVPlacement 时，它的 mother volume 为一个空指针。也不能旋转，而且被置于最开始的全局坐标系中。

总的来说，World volume 最好选择最简单的 solid 来定义，例如 Example N01 中，我们用的 experiment hall 就是一个 world volume.

```
G4VPhysicalVolume* experimentalHall_phys
    = new G4PVPlacement(0,                        // no rotation
                        G4ThreeVector(0.,0.,0.), // translation position
                        experimentalHall_log,     // its logical volume
                        "expHall",                // its name
                        0,                        // its mother volume
                        false,                    // no boolean operations
                        0);                       // its copy number
```

Source listing 2.2.4
The World volume from Example N01.

3）如何定义探测器的材料

总体考虑：

在自然界，一般物质（化合物，混合物）都是由元素构成，而元素由同位素构成。因此，在 Geant4 中有三种主要的类，每一种类都有一个表作为静态数据成员。

G4Element 类用来描述原子的属性：

atomic number,
number of nucleons,
atomic mass,
shell energy,
as well as quantities such as cross sections per atom, etc.

G4Material 类用来描述物质的宏观属性：
density,
state,
temperature,
pressure,
as well as macroscopic quantities like radiation length, mean free path, dE/dx, etc.

G4Material 包含了与最后元素和同位素组成物质的所有信息，同时，还隐藏包含了执行信息。

定义一个简单的物质材料：

在下面这个例子中，通过定义名字，密度，质量/mol，以及原子序数，我们创建了液态氩。

```
G4double density = 1.390*g/cm3;
G4double a = 39.95*g/mole;
G4Material* lAr = new G4Material(name="liquidArgon", z=18., a, density);
```

Source listing 2.3.1
Creating liquid argon.

这个指向材料，lAr 的指针，被用来定义这个物质来创建 logcial volume。

```
G4LogicalVolume* myLbox = new G4LogicalVolume(aBox,lAr,"Lbox",0,0,0);
```

定义一个分子：

下面是个定义水分子的例子，从建立化合物开始，定义分子的原子数。

```
a = 1.01*g/mole;
G4Element* elH  = new G4Element(name="Hydrogen",symbol="H" , z= 1., a);

a = 16.00*g/mole;
G4Element* elO  = new G4Element(name="Oxygen"  ,symbol="O" , z= 8., a);

density = 1.000*g/cm3;
G4Material* H2O = new G4Material(name="Water",density,ncomponents=2);
H2O->AddElement(elH, natoms=2);
H2O->AddElement(elO, natoms=1);
```

Source listing 2.3.2
Creating water by defining its molecular components.

通过质量比来定义混合物

下面这个例子是定义空气混合物，先定义氮氧两种物质，然后给出两种物质的质量比，从而得到空气混合物。

```
a = 14.01*g/mole;
G4Element* elN  = new G4Element(name="Nitrogen",symbol="N" , z= 7., a);

a = 16.00*g/mole;
G4Element* elO  = new G4Element(name="Oxygen"  ,symbol="O" , z= 8., a);

density = 1.290*mg/cm3;
G4Material* Air = new G4Material(name="Air  ",density,ncomponents=2);
Air->AddElement(elN, fractionmass=70*perCent);
Air->AddElement(elO, fractionmass=30*perCent);
```

Source listing 2.3.3
Creating air by defining the fractional mass of its components.

打印输出物质材料信息

```
G4cout << H2O;                                    \\ print a given material
G4cout << *(G4Material::GetMaterialTable());  \\ print the list of materials
```

Source listing 2.3.4
Printing information about materials.

在例子 examples/novice/N03/N03DetectorConstruction.cc 中，你可以看到所有创建物质材料方法的例子。

4）如何定义一个粒子

粒子的定义

Geant4 提供了许多种类型的粒子。

1）ordinary particles, such as electron, proton, and gamma
2）resonant particles with very short life, such as vector mesons, and delta baryons
3）nuclei, such as deuteron, alpha, and heavy ions
4）quarks, di-quarks, and gluons

G4ParticleDefinition这个类提供了以上这些粒子，每个粒子也有它自己从G4ParticleDefinition继承下来的类。

以下是六个主要的粒子：

lepton
meson
baryon
boson
shortlived
ion

这些粒子都在/Geant4/source/particle这个目录里有定义，而且每个粒子都有它对应的库。

G4ParticleDefinition类：

G4ParticleDefinition这个类定义了各个粒子的属性，例如粒子的名字，质量，带电情况，自旋等等。大部分这些属性都是只读的，若不重新编译库是不能被用户改变的。

如何访问一个粒子：

每个粒子类的类型都代表了一个粒子类型，每个类都有一个单一的静态对象。例如：

G4Eletron这个类就代表“电子”，G4Electron:theElectron就是G4Electron唯一的对象。你可以通过调用静态函数G4Electron::ElectronDefinition()来获得“电子”对象的指针。

Geant4默认下大约提供了100种粒子，可以被用于许多物理过程。在一般运用中，用户不必自己来定义粒子。

粒子是各个粒子类的静态对象。这意味着这些对象在main()执行前将自动初始化。然而，在你程序要用到粒子类时，你必须在这个地方对这些类进行宣称，否则编译器将不能识别你所需要的类，粒子类也不能被初始化。

粒子目录：

G4ParticleTable类提供了一个粒子目录，这个目录提供了许多工具方法，例如：

`FindParticle(G4String name)`: find the particle by name

`FindParticle(G4int PDGencoding)`: find the particle by PDG encoding

G4PartileTable也被定义为一个单独的对象，静态函数G4PartileTable:GetParticleTable()给出了它的指针。粒子在构建时自动注册，你不必（也不能）执行注册程序。

阈值：

G4PartileDefinition类在实际的极限范围内有一个唯一的阈值。通过调用每个粒子类的SetCuts()函数，对于在几何体定义中的所有物质材料，用长度表达的阈值就转变为用能量表达的阈值。你可以通过调用GetLengthCuts()来获得长度表达的cut值，调用GetEnergyThreshold(const G4Material *)来获得某一种物质下的极限能量值。

定义粒子和物理过程：

G4VuserPhysicsList类是“用户强制类”中的一个基本类，在其中你必须定义你模拟中将要用到的所有粒子和物理过程。此外，cut-off参数也应该在这个类中定义。

用户必须创建自己从G4VuserPhysicsList继承下来的类，完成以下的纯虚拟函数：

`ConstructParticle()`: construction of particles

`ConstructPhysics()`: construct processes and register them to particles

`SetCuts()`: setting a cut value in range to all particles

在这一章节中有一些简单的ConstructParticle()和SetCuts()例子。

创建粒子：

ConstructParticle函数是一个纯虚拟函数，在函数里，你应该为所有你想要得所有粒子调用它的静态成员函数。

例如，假设你需要一个质子和Geantino（一个用于模拟的虚拟粒子，它不与任何物质发生作用），ConstructParticle函数如以下所示：

```
void ExN01PhysicsList::ConstructParticle()
{
  G4Proton::ProtonDefinition();
  G4Geantino::GeantinoDefinition();
}
```

Source listing 2.4.1
Construct a proton and a geantino.

在Geant4里定义的粒子总数超过100种，若要列出所有粒子是一件很繁琐的事。现在有一些有用类能够在我们需要所有Geant4目录里的粒子时能够调用。这儿提供了六种类对应六

个粒子目录。

G4BosonConstructor
G4LeptonConstructor
G4MesonConstructor
G4BarionConstructor
G4IonConstructor
G4ShortlivedConstructor

可以在 ExN04PhysicsList 中看到一个例子：

```
void ExN04PhysicsList::ConstructLeptons()
{
  // Construct all leptons
  G4LeptonConstructor pConstructor;
  pConstructor.ConstructParticle();
}
```

Source listing 2.4.2
Construct all leptons.

设置阈值：

SetCuts()方法是一个纯虚拟函数。你应该通过调用每个粒子类的 SetCuts()方法来为每个粒子设置阈值。粒子，几何体以及物理过程的建立都应该处理调用 SetCuts()。用长度表达的阈值是唯一的，这是 Geant4 的一个重要特性。对于通常的运用，用户需要决定一个长度阈值，然后对所用的粒子都采用这个阈值。在这种情况下，你可以用 SetCutsWithDefault()这个方法，它是 G4VuserPhysicsList 类提供的，具有一个 defaultCutValue 成员作为默认阈值。如下：

```
void ExN04PhysicsList::SetCuts()
{
  //    the G4VUserPhysicsList::SetCutsWithDefault() method sets
  //    the default cut value for all particle types
  SetCutsWithDefault();
}
```

Source listing 2.4.3
Set cut values by using the default cut value.

这个 defaultCutValue 默认设为 1.0mm。当然，你也可以在 physics list 建立一个新的默认阈值。可以通过调用 G4VuserPhysicsList 里的 SetDefaultCutValue()：

```
ExN04PhysicsList::ExN04PhysicsList():  G4VUserPhysicsList()
{
  // default cut value  (1.0mm)
  defaultCutValue = 1.0*mm;
}
```

Source listing 2.4.4
Set the default cut value.

假如你想对于不同的粒子设置不同的阈值，你在设置阈值时需要知道粒子类型的顺序，这是因为在计算截面时，一些粒子需要其他粒子类型的阈值。这个粒子顺序为：

1. gamma
2. electron
3. positron
4. proton and antiproton
5. others

为了简便SetCuts()方法的执行，G4VuserPhysicsList类提供了一些方法：

```
SetCutValue(G4double cut_value, G4String particle_name)
SetCutValueForOthers(G4double cut_value)
SetCutValueForOtherThan(G4double cut_value, G4ParticleDefinition*
a_particle)
```

下面是SetCuts()的一个例子的执行：

```
void ExN03PhysicsList::SetCuts()
{
  // set cut values for gamma at first and for e- second and next for e+,
  // because some processes for e+/e- need cut values for gamma
  SetCutValue(cutForGamma, "gamma");
  SetCutValue(cutForElectron, "e-");
  SetCutValue(cutForElectron, "e+");

  // set cut values for proton and anti_proton before all other hadrons
  // because some processes for hadrons need cut values for proton/anti_proton
  SetCutValue(cutForProton, "proton");
  SetCutValue(cutForProton, "anti_proton");

  SetCutValueForOthers(defaultCutValue);
}
```

Source listing 2.4.5
Example implementation of the SetCuts() method.

你不能在事件循环中改变阈值。你可以通过利用用户命令/run/particle/SetCuts在一次运行run后改变阈值。

5）如何定义一个物理过程

物理过程描述的是如何与物质材料和粒子之间的相互作用。各种各样的电磁作用，强子作用和其它相互作用都在Geant4里有定义。在这儿列出7种主要作用：

electromagnetic
hadronic

transportation
decay
optical
photolepton_hadron
parameterisation

G4VProcess 被提供作为物理过程的一个基类。每个类通过利用三个 DoIt 函数来描述它的行为：

AtRestDoIt

AlongStepDoIt

PostStepDoIt

管理过程：

G4ProcessManager 类包含了一列一个粒子能承担的所有过程。它含有过程调用顺序的信息。一个 G4ProcessManager 对象对应着每一个粒子，附在 G4ParticleDefinition 类里。为了使过程生效，过程应该被注册到粒子的 G4ProcessManager 中，通过 AddProcess()和 SetProcessOrdering()函数来调整包含的信息。

在运行期间，通过利用 ActivateProcess() 和 InActivateProcess() 函数，G4ProcessManager()具有功能将一些过程激化或使之无作用。

定义物理过程：

G4VusePhysicsList 类是属于“user mandatory class”的基类，在其中你必须和你模拟中将用到的所有粒子一起定义所有的物理过程。用户必须创建自己从 G4VuserPhysicsList 中继承下来的类，然后执行这个纯虚拟函数 ConstructPhysics()。G4VusePhysicsList 类创建 G4ProcessManger 对象，并把这些对象附加给所有定义在函数 ConstructParticle()的粒子类中。

加入一个运输方法：

G4Transportation类应该被注册到所有粒子类当中，因为没有一个粒子运输过程，粒子就不能被跟踪，这个过程是描述空间和时间上的移动。AddTransportation()函数在 G4VusePhysicsList类中被提供，因此它必须在函数ConstructPhysics()中被调用。

```
void G4VUserPhysicsList::AddTransportation()
{
  // create G4Transportation
  G4Transportation* theTransportationProcess = new G4Transportation();

  // loop over all particles in G4ParticleTable and register the transportation pr
  theParticleIterator->reset();

  while( (*theParticleIterator)() ){
    G4ParticleDefinition* particle = theParticleIterator->value();
    G4ProcessManager* pmanager = particle->GetProcessManager();

    // adds transportation to all particles except shortlived particles
    if (!particle->IsShortLived()) {
      pmanager ->AddProcess(theTransportationProcess);
      // set ordering to the first for AlongStepDoIt
      pmanager ->SetProcessOrderingToFirst(theTransportationProcess, idxAlongStep)
    }
  }
}
```

Source listing 2.5.1
Add a transportation method.

创建和注册物理过程：

ConstructProcess()是一个用于创建物理过程并把它们注册给粒子的纯虚拟函数。例如，假如你仅仅采用粒子的 G4Geantino 类，那么仅仅就 geantino 承担运输过程。此时的 ConstructProcess()函数如下：

```
void ExN01PhysicsList::ConstructProcess()
{
  // Define transportation process
  AddTransportation();
}
```

Source listing 2.5.2
Register processes for a geantino.

对于伽马粒子来说，电磁过程的注册实现如下：

```
void MyPhysicsList::ConstructProcess()
{
  // Define transportation process
  AddTransportation();
  // electromagnetic processes
  ConstructEM();
}
void MyPhysicsList::ConstructEM()
{
   //  Get the process manager for gamma
  G4ParticleDefinition* particle = G4Gamma::GammaDefinition();
  G4ProcessManager* pmanager = particle->GetProcessManager();

  // Construct processes for gamma
  G4PhotoElectricEffect * thePhotoElectricEffect = new G4PhotoElectricEffect();
  G4ComptonScattering * theComptonScattering = new G4ComptonScattering();
  G4GammaConversion* theGammaConversion = new G4GammaConversion();

  // Register processes to gamma's process manager
  pmanager->AddDiscreteProcess(thePhotoElectricEffect);
  pmanager->AddDiscreteProcess(theComptonScattering);
  pmanager->AddDiscreteProcess(theGammaConversion);
}
```

Source listing 2.5.3
Register processes for a gamma.

对于其它过程和粒子而言，在 G4ProcessManager 中注册是一个复杂的程序，因为过程与过程之间的关系是至关重要的。

6）如何创建一个主事例

创建主事例：

G4VuserPrimaryGeneratorAction 是强制类之一，它从你自己的具体的类继承而来。在你具体类中，你必须定义如何创建一个主事例。实际主事例的创建是通过 G4VprimaryGenerator 的具体类得到的，这在下一节会具体解释。G4VuserPrimaryGeneratorAction 具体类将定义主粒子创建的方法。

```
#ifndef ExN01PrimaryGeneratorAction_h
#define ExN01PrimaryGeneratorAction_h 1

#include "G4VUserPrimaryGeneratorAction.hh"

class G4ParticleGun;
class G4Event;

class ExN01PrimaryGeneratorAction : public G4VUserPrimaryGeneratorAction
{
  public:
    ExN01PrimaryGeneratorAction();
    ~ExN01PrimaryGeneratorAction();

  public:
    void generatePrimaries(G4Event* anEvent);

  private:
    G4ParticleGun* particleGun;
};

#endif

#include "ExN01PrimaryGeneratorAction.hh"
#include "G4Event.hh"
#include "G4ParticleGun.hh"
#include "G4ThreeVector.hh"
#include "G4Geantino.hh"
#include "globals.hh"

ExN01PrimaryGeneratorAction::ExN01PrimaryGeneratorAction()
{
  G4int n_particle = 1;
  particleGun = new G4ParticleGun(n_particle);

  particleGun->SetParticleDefinition(G4Geantino::GeantinoDefinition());
  particleGun->SetParticleEnergy(1.0*GeV);
  particleGun->SetParticlePosition(G4ThreeVector(-2.0*m,0.0*m,0.0*m));
}

ExN01PrimaryGeneratorAction::~ExN01PrimaryGeneratorAction()
{
  delete particleGun;
}

void ExN01PrimaryGeneratorAction::generatePrimaries(G4Event* anEvent)
{
  G4int i = anEvent->get_eventID() % 3;
  switch(i)
  {
    case 0:
      particleGun->SetParticleMomentumDirection(G4ThreeVector(1.0,0.0,0.0));
      break;
    case 1:
      particleGun->SetParticleMomentumDirection(G4ThreeVector(1.0,0.1,0.0));
      break;
    case 2:
      particleGun->SetParticleMomentumDirection(G4ThreeVector(1.0,0.0,0.1));

      break;
  }

  particleGun->generatePrimaryVertex(anEvent);
}
```

Source listing 2.6.1
An example of a *G4VUserPrimaryGeneratorAction* concrete class using *G4ParticleGun*.

一个事件的创建：

G4VuserPrimaryGeneratorAction 有一个纯虚拟函数名为 generatePrimaries()。这个函数在每个事件的开头调用。在这个函数中，必须通过 generatePrimaryVertex()函数来调用声明的 G4VuserPrimaryGenerator 具体类。

G4VprimaryGenerator:

Geant4 提供了两个 G4VprimaryGenerator 具体的类。一个是 G4ParticleGun，另一个为 G4HEPEvtInterface。

G4ParticleGun:

G4ParticleGun 是 Geant4 提供的一个创建器。这个类主要在指定的时刻和位置创建发射粒子。这个类不提供任何随机化的种类。

在实际中，随机创建一个主发射粒子的能量，时刻或位置是相当常见的。这时，可以通过调用由 G4ParticleGun 提供的不同方法来实现。这些方法的调用应该在 geaneratePrimaries()函数中实现，并且在 G4ParticleGun 的 genaeratePrimaryVertex()之前调用。Geant4 提供了许多随机数创建的方法。

G4ParticleGun 的公共函数：

以下为 G4ParticleGun 提供的函数，所有这些函数都能被从 geanratePrimaries()中调用。

```
void SetParticleDefinition(G4ParticleDefinition*)
void SetParticleMomentum(G4ParticleMomentum)
void SetParticleMomentumDirection(G4ThreeVector)
void SetParticleEnergy(G4double)
void SetParticleTime(G4double)
void SetParticlePosition(G4ThreeVector)
void SetParticlePolarization(G4ThreeVector)
void SetNumberOfParticles(G4int)
```

7）如何编译成一个可执行程序

在 Unix 环境中编译 ExampleN01：

这个用户例子程序的代码在位于目录$G4INSTALL/examples/novice 中。在以下小结中，我们将大概的浏览一下 Geant4 中 GNUmake 的工作机制。我们还将说明如何编译一个具体的例子，“ExamplesN01”，这是 Geant4 发布的一部分。

Geant4 中 GNUmake 如何工作：

GNUmake 过程主要受以下的 GNUmake 脚本文件控制(*.gmk 脚本位于 $G4INSTALL/config 目录下)

Architecture.gmk：调用和定义所有具体设置和路径的结构，它们存储在 $G4INSTALL/config/sys 中。

Globlib.gmk：为编译库文件定义 GNUmake 规则。

Common.gmk：为编译对象和库定义 GNUmake 规则。

Binmake.gmk：为编译成可执行文件定义 GNUmake 规则。

GNUmakefile：放置于发布的 Geant4 中的每个目录。

内核库默认下放在$G4INSTALL/lib/$G4SYSTEM 下，这里的$G4SYSTEM 定义了系统结构和正在使用的编译器。可执行的二进制文件放在$G4WORKDIR/bin/$G4SYSTEM 下，暂时文件在$G4WORKDIR/tmp/$G4SYSTEM 下。$G4WORKDIR 应该由用户定义其位置来放置自己在 Geant4 下的工作目录。

编译成执行文件：

编译成一个可执行文件，例如来自发布中的一个例子是从调用“gmake”命令开始。以exampleN01为例，命令如下：

```
> cd $G4WORKDIR/examples/novice/N01
> gmake
```

这样在$G4WORKDIR/bin/$G4SYSTEM 中将产生一个“exampleN01”的可执行文件，你可以直接调用和执行它。实际上应该把$G4WORKDIR/bin/$G4SYSTEM 加入环境路径设置中。

8）如何执行程序

介绍：

- 纯代码运行批模式
- 批模式，但通过读取宏命令
- 交互模式，由命令行执行

a) 纯代码批模式：

以下为一个应用程序主程序的示例，它是以批模式方式运行：

```
int main()
{
  // Construct the default run manager
  G4RunManager* runManager = new G4RunManager;

  // set mandatory initialization classes
  runManager->SetUserInitialization(new ExN01DetectorConstruction);
  runManager->SetUserInitialization(new ExN01PhysicsList);

  // set mandatory user action class
  runManager->SetUserAction(new ExN01PrimaryGeneratorAction);

  // Initialize G4 kernel
  runManager->Initialize();

  // start a run
  int numberOfEvent = 1000;
  runManager->BeamOn(numberOfEvent);

  // job termination
  delete runManager;
  return 0;
}
```

Source listing 2.9.1
An example of the `main()` routine for an application which will run in batch mode.

在 main()中运行的事件数已被固定住，若要改变这个数字，至少需要再次编译 main()。

b) 用宏文件的批模式：

以下为一个应用程序主程序的示例，它是以批模式方式运行，但是通过读取一个命令文件：

```
int main(int argc,char** argv) {

   // Construct the default run manager
  G4RunManager * runManager = new G4RunManager;

  // set mandatory initialization classes
  runManager->SetUserInitialization(new MyDetectorConstruction);
  runManager->SetUserInitialization(new MyPhysicsList);

  // set mandatory user action class
  runManager->SetUserAction(new MyPrimaryGeneratorAction);

  // Initialize G4 kernel
  runManager->Initialize();

  //read a macro file of commands
  G4UImanager * UI = G4UImanager::getUIpointer();
  G4String command = "/control/execute ";
  G4String fileName = argv[1];
  UI->applyCommand(command+fileName);

  delete runManager;
  return 0;
}
```

Source listing 2.9.2
An example of the `main()` routine for an application which will run in batch mode, but reading a file of commands.

这个例子将通过如下命令执行：

```
> myProgram run1.mac
```

在这里，myprogram 是执行文件的文件名，run1.mac 为命令宏，位于当前目录下，它可能为如下内容：

```
#
# Macro file for "myProgram.cc"
#
# set verbose level for this run
#
/run/verbose       2
/event/verbose     0
/tracking/verbose  1
#
# Set the initial kinematic and run 100 events
# electron 1 GeV to the direction (1.,0.,0.)
#
/gun/particle e-
/gun/energy 1 GeV
/run/beamOn 100
```

Source listing 2.9.3
A typical command macro.

事实上，你可以在不同的运行条件下重新执行你的程序，而不需要重新编译任何东西。

c) 通过命令行执行的交互模式

以下为一个应用程序主程序的示例，它是以交互的方式运行，它正在等待来自键盘输入的命令行：

```
int main(int argc,char** argv) {

  // Construct the default run manager
  G4RunManager * runManager = new G4RunManager;

  // set mandatory initialization classes
  runManager->SetUserInitialization(new MyDetectorConstruction);
  runManager->SetUserInitialization(new MyPhysicsList);

  // visualization manager
  G4VisManager* visManager = new MyVisManager;
  visManager->Initialize();

  // set user action classes
  runManager->SetUserAction(new MyPrimaryGeneratorAction);
  runManager->SetUserAction(new MyRunAction);
  runManager->SetUserAction(new MyEventAction);
  runManager->SetUserAction(new MySteppingAction);

  // Initialize G4 kernel
  runManager->Initialize();

  // Define UI terminal for interactive mode
  G4UIsession * session = new G4UIterminal;
  session->SessionStart();
  delete session;

  // job termination
  delete visManager;
  delete runManager;

  return 0;
}
```

这个例子首先执行可执行文件：

➢ myprogram

G4 内核将立即提示：

Idle>

然后你就可以开始你的命令。例如发出创建空场景("world"为默认)

```
Idle> /vis/scene/create
```

给空场景内加入几何体

```
Idle> /vis/scene/add/volume
```

创建一个观察器

```
Idle> /vis/viewer/create
```

画出整个场景

```
Idle> /vis/scene/notifyHandlers
Idle> /run/verbose 0
Idle> /event/verbose 0
Idle> /tracking/verbose 1
Idle> /gun/particle mu+
Idle> /gun/energy 10 GeV
Idle> /run/beamOn 1
```

```
Idle> /gun/particle proton
Idle> /gun/energy 100 MeV
Idle> /run/beamOn 3
Idle> exit
```

9）如何使探测器和事件可视化

介绍：

在这里我们简单的解释如何完成 Geant4 的可视化。这里的描述都是基于例子程序 examples/novice/No3 的。

可视化驱动：

Geant4 的可视化必须对不同的用户需求做出反应。但只有一个内置观察器要满足所有要求是困难的，因此 Geant4 的可视化定义了一个抽象的接口来满足任意多的图像系统。在这儿的图像系统是指与 Geant4 无关的作为一个进程的应用程序，或是由 Geant4 编译的图像库文件。Geant4 的可视化发布支持与几个图像系统的具体接口，它们在许多方面互相补充。与一个图像系统具体的接口被称为“可视化驱动”。

用户不必用所有的可视化驱动。可以选择那些适合需求的部分。在下面，处于简单考虑，我们假设 Geant4 的库由“DAWNFILE 驱动”和“OpenGL-Xlib 驱动”联合建立的。

安装可视化驱动建立 Geant4 库时的注意事项：

为了完成 Geant4 的可视化，用户必须为他所选择的可视化驱动建立 Geant4 库文件。可视化驱动的选择可以手工或是由 GNUmakefile 设置正确地 C 预处理标志来得到。

每个可视化驱动都有自己的 C 预处理标志。例如，DAWNFILE 驱动通过设置标志 G4VIS_BUILD_DAWNFILE_DRIVER 为 1 来选择。除此之外，假如你想选择不止一个可视化驱动，那么一个全局 C 预处理标志 G4VIS_BUILD 也必须设为 1。

在建立Geant4库文件时，用户可以通过设置环境变量G4VIS_BUILD_DRIVERNAME_DRIVER 为 1 来组合所选择的可视化驱动，这里的 DRIVERNAME 是所选择的可视化驱动的名字。

举个例子，假如你想选择组合 DAWNFILE 和 OPENGL-XLIB 驱动融入 Geant4 库文件当中，环境为 UNIX 的 C shell，在建立库文件爱之前，环境变量可以如下设置：

```
% setenv G4VIS_BUILD_DAWNFILE_DRIVER 1
% setenv G4VIS_BUILD_OPENGLX_DRIVER 1
```

在执行中如何实现可视化驱动：

用户可以在 Geant4 的执行中实现可视化驱动，当然前提是在 Geant4 中安装了驱动的库文件。在标准发布中，每个可视化驱动都能如下一样容易实现。

在这个例子中，你将发现用户可视化管理器等。如 examples/novice/N03/include/ExN03VisManager.hh和src/ExN03VisManager.cc. 假如用户采用这种形式，那么所有你需要的C预处理标志都应该手工或GNUmakefile提供设置。每个可视化驱动都有自己的C预处理标志。例如，DAWNFILE驱动就是通过设置 G4VIS_USE_DAWNFILE为1。除此之外，有一个全局的C预处理标志G4VIS_USE，若用户想在Geant4实现不止一种可视化驱动，这个标志就必须设为1。

用户可以通过设置环境变量实现可视化驱动，它的名字形式为 G4VIS_USE_DRIVERNAME，把它设为1。这里的DRIVERNAME是驱动的名字。例如，用户若想实现DAWNFILE和OPENGL_Xlib这两个驱动的执行，环境在Unix的C shell下，编译前变量如下设置：

```
% setenv G4VIS_USE_DAWNFILE 1
% setenv G4VIS_USE_OPENGLX 1
```

5. 其它

从Geant4的应用过程中，我们发现它不仅可以用来进行粒子物理探测器模拟工作，还能在许多与放射性应用相关的领域中广泛应用，例如核医学、生物科学、辐射防护学等等。同时它还可以在放射性材料科学中得到应用，比如按照一定的材料组分来模拟其辐照特性就可以为抗辐照材料或辐射敏感材料的研究生产提供宝贵的资料。

B. Geant4 系统中基于闪烁光纤阵列的 CT 扫描程序

DetectorConstruction.cc

```
// $Id: ExpDetectorConstruction.cc,v 1.10 2002/11/05 Ma Qing-li
// GEANT4 tag $Name: geant4-04-00 $
//  ......oooOOOOOooo........oooOOOOOooo......
#include "ExpDetectorMessenger.hh"
#include "ExpScintSD.hh"
#include "ExpDetectorConstruction.hh"
#include "G4TransportationManager.hh"
#include "G4Tubs.hh"
#include "G4Material.hh"
#include "G4Box.hh"
#include "G4LogicalVolume.hh"
#include "G4PVPlacement.hh"
#include "G4PVParameterised.hh"
#include "G4SDManager.hh"
#include "G4VisAttributes.hh"
#include "G4Colour.hh"
#include "G4ios.hh"
#include "G4RunManager.hh"
#include "G4VVisManager.hh"
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOO
OOooo......

ExpDetectorConstruction::ExpDetectorConstruction()
:solidWorld(0),  logicWorld(0),  physiWorld(0),
 solidSource(0), logicSource(0), physiSource(0),
 solidObject1(0),logicObject1(0),Object1_phys(0),
 solidObject2(0),logicObject2(0),Object2_phys(0),
 solidObject3(0),logicObject3(0),Object3_phys(0),
 solidFiber(0),  logicFiber(0),  Fiber_phys(0),
 solidShield(0), logicShield(0), physiShield(0),
 Object1Mater(0),Object2Mater(0),Object3Mater(0),
FiberMater(0),aScintSD(0)
//
fWorldLength(0),SFD(0),SOD(0),fSourceLength(0),ObjectLength(0),FiberR
adius(0),theta(0),
//Objectchanged(false)

{
  SFD= 30*cm;                 // Source to Object Distance
  SOD= 28*cm;                 // Source to Fiber Distance
  FiberLength =12.0*cm;
  FiberDiameter = 0.025*cm;
  FiberRadius = 0.5*FiberDiameter;

  fSourceLength = 0.5*cm;
  ObjectLength  = 0.5*cm;      // Half length of the Object
  theta=0;
 G4int CoNo;
//ComputeParameters();
detectorMessenger = new ExpDetectorMessenger(this);
```

```
}

//....oooOOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOO
OOooo......

ExpDetectorConstruction::~ExpDetectorConstruction()
{
 delete detectorMessenger;
}
//....oooOOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOO
OOooo......
G4VPhysicalVolume* ExpDetectorConstruction::Construct()
{
DefineMaterials();
  return ConstructCT();
}


void ExpDetectorConstruction::DefineMaterials()
{
 G4double a, iz, z, density;
  G4String name, symbol;
//  G4double temperature, pressure;
  G4int nel;

  //Air
    a = 14.01*g/mole;
    G4Element* elN = new G4Element(name="Nitrogen", symbol="N", iz=7.,
a);
    a = 16.00*g/mole;
    G4Element* elO = new G4Element(name="Oxigen", symbol="O", iz=8., a);
    density = 1.29*mg/cm3;
    G4Material* Air = new G4Material(name="Air", density, nel=2);
    Air->AddElement(elN, .7);
    Air->AddElement(elO, .3);

 //Cu

  density = 8.960*g/cm3;
  a = 63.55*g/mole;
  G4Material* Cu = new G4Material(name="Copper"   , z=29., a, density);

 //Pb
    a = 207.19*g/mole;
    density = 11.35*g/cm3;
    G4Material* Pb = new G4Material(name="Pb", z=82., a, density);

  //Al
    a = 26.98*g/mole;
    density = 2.7*g/cm3;
    G4Material* Al = new G4Material(name="Aluminum", z=13., a, density);

  //SiO2
    G4Element* elSi = new G4Element("Silicon","Si",14.,28.0855*g/mole);
    G4Material* SiO2 = new G4Material("SiO2",  2.2*g/cm3, 2, kStateSolid,
293.0*kelvin, 1.0*atmosphere );
   SiO2->AddElement( elO, 2 );
   SiO2->AddElement( elSi, 1 );
```

```
  //Fiber
    G4Element*elH = new G4Element("Hydrogen","H",1.,1.00794*g/mole);
    G4Element*elC = new G4Element("Carbon","C",6.,12.011*g/mole);
    G4Material*Polysthyrene = new
G4Material("Polysthyrene",1.032*g/cm3,2,kStateSolid,293.0*kelvin,1.0*
atmosphere);
    Polysthyrene->AddElement(elH,8);
    Polysthyrene->AddElement(elC,8);
 // Print all the materials defined.
  //
    G4cout << G4endl << "The materials defined are : " << G4endl << G4endl;
    G4cout << *(G4Material::GetMaterialTable()) << G4endl;
  const G4int NUMENTRIES = 2;
  G4double Polysthyrene_PP[NUMENTRIES]    = { 3.0*eV,3.0*eV};
  G4double Polysthyrene_SCINT[NUMENTRIES] = { 1.0,1.0};
  G4double Polysthyrene_RIND[NUMENTRIES]  = { 1.6,1.6};
  G4double Polysthyrene_ABSL[NUMENTRIES]  = { 42.*cm,42.*cm};
  G4MaterialPropertiesTable *Polysthyrene_mt = new
G4MaterialPropertiesTable();
  Polysthyrene_mt->AddProperty("SCINTILLATION", Polysthyrene_PP,
Polysthyrene_SCINT, NUMENTRIES);
  Polysthyrene_mt->AddProperty("RINDEX",       Polysthyrene_PP,
Polysthyrene_RIND,  NUMENTRIES);
  Polysthyrene_mt->AddProperty("ABSLENGTH",    Polysthyrene_PP,
Polysthyrene_ABSL,  NUMENTRIES);
  Polysthyrene->SetMaterialPropertiesTable(Polysthyrene_mt);

  WorldMater=Air;
  FiberMater = Polysthyrene;
  Object1Mater = Al;
  Object2Mater = Al;
  Object3Mater = Al;
  ShieldMater=Pb;
 }

G4VPhysicalVolume* ExpDetectorConstruction::ConstructCT()
{
//ComputeParameters();

//--------- Definitions of Solids, Logical Volumes, Physical Volumes
---------

//------------------------------
// World
//------------------------------


  fWorldLength= 2*(SFD+FiberLength);
  G4double HalfWorldLength = 0.5*fWorldLength;
  G4ThreeVector positionWorld = G4ThreeVector(0,0,0);

if(solidWorld) delete solidWorld;
if(logicWorld) delete logicWorld;
if(physiWorld) delete physiWorld;

  solidWorld= new
G4Box("world",HalfWorldLength,HalfWorldLength,HalfWorldLength);
```

```
  logicWorld= new G4LogicalVolume( solidWorld, WorldMater, "World", 0,
0, 0);
  physiWorld = new G4PVPlacement(0,              // no rotation
                            G4ThreeVector(), // at (0,0,0)
                "World",        // its name
                            logicWorld,      // its logical volume
                            0,               // its mother  volume
                            false,           // no boolean operations
                            CoNo);           // no field specific to
volume

//------------------------------
// Source
//------------------------------

  G4double HalfSourceLength = 0.5*fSourceLength;
  G4ThreeVector positionSource = G4ThreeVector(0,0,0);
  solidSource= new
G4Box("Source",HalfSourceLength,HalfSourceLength,HalfSourceLength+2);
  logicSource= new G4LogicalVolume( solidSource, WorldMater, "Source",
0, 0, 0);
  physiSource = new G4PVPlacement(0,             // no rotation
                           G4ThreeVector(), // at (0,0,0)
                "Source",       // its name
                            logicSource,     // its logical volume
                            physiWorld,      // its mother  volume
                            false,           // no boolean operations
                            0);              // no field specific to volume

////////////////////////////////////////////////////////////////////////
////////////////////////

//------------------------------
// Object1(yellow)
//------------------------------


{
  solidObject1 = new G4Tubs("Object1",
0.0*cm,0.0125*cm,0.5*ObjectLength ,0.0*deg,360.0*deg);
  logicObject1 = new
G4LogicalVolume(solidObject1,Object1Mater,"Object1",0,0,0);

  G4double xo, yo, zo;
    yo = 0.0;
    xo = 0.0;
    zo = SOD+(0.5*ObjectLength)*cos(theta);
    G4RotationMatrix rmo;
    rmo.rotateY(theta);
    Object1_phys
      = new G4PVPlacement(G4Transform3D(rmo,G4ThreeVector(xo,yo,zo)),
                          "Object1",logicObject1,
                          physiWorld,false,0);



//------------------------------
// Object2(purple)
```

```
// Object2(purple)
//------------------------------

  solidObject2 = new
G4Tubs("Object2",0.5*cm,0.8*cm,ObjectLength,0.0*deg,360.0*deg);
  logicObject2 = new
G4LogicalVolume(solidObject2,Object2Mater,"Object2",0,0,0);
  Object2_phys
     = new G4PVPlacement(G4Transform3D(rmo,G4ThreeVector(xo,yo,zo)),
                         "Object2",logicObject2,
                         physiWorld,false,0);
//------------------------------
// Object3(yellow)
//------------------------------

  solidObject3 = new G4Tubs("Object3",
0.9*cm,1.1*cm,0.5*ObjectLength,0.0*deg,360.0*deg);
  logicObject3 = new
G4LogicalVolume(solidObject3,Object3Mater,"Object3",0,0,0);
  Object3_phys
     = new G4PVPlacement(G4Transform3D(rmo,G4ThreeVector(xo,yo,zo)),
                         "Object3",logicObject3,
                         physiWorld,false,0);
}
////////////////////////////////////////////////////////////////////////
//////////////////////////
//------------------------------
// (Fiber)
//------------------------------


  G4ThreeVector positionFiber = G4ThreeVector(0,0,SFD);
  solidFiber = new G4Tubs("Fiber",
0.0*cm,FiberRadius,0.5*FiberLength ,0.0*deg,360.0*deg);
  logicFiber = new
G4LogicalVolume(solidFiber,FiberMater,"Fiber",0,0,0);

    G4int ii;
    G4float phi, xf, yf, zf;
    G4float tg;
    tg=atan(FiberDiameter/SFD);
  for(ii=-50 ;ii<=50 ;ii++)
  {
    phi =(tg)*ii;
    xf = (SFD+(0.5*FiberLength))*sin(phi);
    yf = 0;
    zf = (SFD+(0.5*FiberLength));
    G4RotationMatrix rm;
    rm.rotateY(phi);
    Fiber_phys
      = new G4PVPlacement(G4Transform3D(rm,G4ThreeVector(xf,yf,zf)),
                         "Fiber",logicFiber,
                         physiWorld,false,ii);
  }

//-------------------------------------------------
//Shielding
//-------------------------------------------------
```

```
  G4ThreeVector positionShield =
G4ThreeVector(0,0,SFD+(0.5*FiberLength));
  solidShield= new G4Box("Shield",2.0*cm,0.1*cm,0.49*FiberLength);
  logicShield= new G4LogicalVolume( solidShield, ShieldMater, "Shield",
0, 0, 0);
  physiShield = new G4PVPlacement(0,
                                  positionShield,
                  "Shield",       // its name
                                  logicShield,     // its logical volume
                                  physiWorld,      // its mother  volume
                                  false,           // no boolean operations
                                  0);              // no field specific to
volume

//------------------------------------------------
// Sensitive detectors
//------------------------------------------------

  G4SDManager* SDman = G4SDManager::GetSDMpointer();
if(!aScintSD)
{
  G4String sensname="CTSD";
//  ExpScintSD* aScintSD = new ExpScintSD("sensname",this);
  aScintSD = new ExpScintSD("sensname",this);
  SDman->AddNewDetector(aScintSD);
}
if(logicFiber)
  logicFiber->SetSensitiveDetector(aScintSD);

//--------- Visualization attributes -------------------------------

  G4VisAttributes* BoxVisAtt= new
G4VisAttributes(G4Colour(1.0,1.0,1.0));
  logicWorld  ->SetVisAttributes(G4VisAttributes::Invisible);
//  logicWorld  ->SetVisAttributes(BoxVisAtt);
  logicSource  ->SetVisAttributes(BoxVisAtt);
 G4VisAttributes* Box1VisAtt= new
G4VisAttributes(G4Colour(0.0,0.0,1.0));
 logicShield  ->SetVisAttributes(Box1VisAtt);


  G4VisAttributes* Tubs0VisAtt = new
G4VisAttributes(G4Colour(1.0,1.0,1.0));
  logicFiber->SetVisAttributes(Tubs0VisAtt);
  G4VisAttributes* Tubs1VisAtt = new
G4VisAttributes(G4Colour(1.0,0.0,0.0));
  logicObject1->SetVisAttributes(Tubs1VisAtt);
  G4VisAttributes* Tubs2VisAtt = new
G4VisAttributes(G4Colour(1.0,0.0,1.0));
  logicObject2->SetVisAttributes(Tubs2VisAtt);
  G4VisAttributes* Tubs3VisAtt = new
G4VisAttributes(G4Colour(1.0,1.0,0.0));
  logicObject3->SetVisAttributes(Tubs3VisAtt);


  return physiWorld;
}
```

```
//....ooo00000ooo.....ooo00000ooo.....ooo00000ooo.....ooo000
00ooo.....

void ExpDetectorConstruction::setWorldMaterial(G4String materialName)
{
  // search the material by its name
  G4Material* pttoMaterial = G4Material::GetMaterial(materialName);
  if (pttoMaterial)
     {Object1Mater = pttoMaterial;
      logicWorld->SetMaterial(pttoMaterial);

     }
}
//....ooo00000ooo.....ooo00000ooo.....ooo00000ooo.....ooo000
00ooo.....

void ExpDetectorConstruction::setObject1Material(G4String
materialName)
{
  // search the material by its name
  G4Material* pttoMaterial = G4Material::GetMaterial(materialName);
  if (pttoMaterial)
     {Object1Mater = pttoMaterial;
      logicObject1->SetMaterial(pttoMaterial);
//      G4cout << "\n----> The Object is " << ObjectLength/cm << " cm of
"
//             << materialName << G4endl;
     }
}
//....ooo00000ooo.....ooo00000ooo.....ooo00000ooo.....ooo000
00ooo.....
void ExpDetectorConstruction::setObject2Material(G4String
materialName)
{
  // search the material by its name
  G4Material* pttoMaterial = G4Material::GetMaterial(materialName);
  if (pttoMaterial)
     {Object2Mater = pttoMaterial;
      logicObject2->SetMaterial(pttoMaterial);
//      G4cout << "\n----> The Object is " << ObjectLength/cm << " cm of
"
//             << materialName << G4endl;
     }
}
//....ooo00000ooo.....ooo00000ooo.....ooo00000ooo.....ooo000
00ooo.....
void ExpDetectorConstruction::setObject3Material(G4String
materialName)
{
  // search the material by its name
  G4Material* pttoMaterial = G4Material::GetMaterial(materialName);
  if (pttoMaterial)
     {Object3Mater = pttoMaterial;
      logicObject3->SetMaterial(pttoMaterial);
//      G4cout << "\n----> The Object is " << ObjectLength/cm << " cm of
"
//             << materialName << G4endl;
     }
```

```
}
//....oooOOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOO
OOooo.....

void ExpDetectorConstruction::setFiberMaterial(G4String materialName)
{
  // search the material by its name
  G4Material* pttoMaterial = G4Material::GetMaterial(materialName);
  if (pttoMaterial)
     {FiberMater = pttoMaterial;
      logicFiber->SetMaterial(pttoMaterial);
      G4cout << "\n----> The Fibers are " << FiberLength/cm << " cm of "
             << materialName << G4endl;
     }
}
//....oooOOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOO
OOooo.....

void ExpDetectorConstruction::setShieldMaterial(G4String materialName)
{
  // search the material by its name
  G4Material* pttoMaterial = G4Material::GetMaterial(materialName);
  if (pttoMaterial)
     {ShieldMater = pttoMaterial;
      logicShield->SetMaterial(pttoMaterial);

     }
}

//....oooOOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOO
OOooo.....
void ExpDetectorConstruction::Settheta(G4double Value)
{
//  Objectchanged=false;
//  ComputeParameters();
  theta=Value;
// theta=theta+0.0175;
}


//....oooOOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOO
OOooo.....

void ExpDetectorConstruction::UpdateGeometry()
{

//rotation under 2 degree
 theta=theta+0.034906585;

 G4RunManager* theRunManager=G4RunManager::GetRunManager();
 theRunManager->DefineWorldVolume(ConstructCT());
 geometryInitialized = true;
 G4cout<<theta<<G4endl;
//ConstructCT();

 G4TransportationManager::GetTransportationManager()
    ->GetNavigatorForTracking()
    ->SetWorldVolume(physiWorld);
```

```
  // Let VisManager know it
  G4VVisManager* pVVisManager = G4VVisManager::GetConcreteInstance();
  if(pVVisManager) pVVisManager->GeometryHasChanged();

  geometryNeedsToBeClosed = true;

}
//....oooOOOOOOooo.....oooOOOOOOooo.....oooOOOOOOooo.....oooOOOOOOooo.....
```

PhysicsList.cc

```
// $Id: ExpPhysicsList.cc,v 1.10 2002/11/05 Ma Qing-li
// GEANT4 tag $Name: geant4-04-00 $

#include "ExpPhysicsList.hh"
//#include "ExpPhysicsListMessenger.hh"
#include "ExpDetectorConstruction.hh"

#include "G4ParticleDefinition.hh"
#include "G4ParticleWithCuts.hh"
#include "G4ProcessManager.hh"
#include "G4ParticleTypes.hh"
#include "G4ParticleTable.hh"
#include "G4ios.hh"
#include "ExpDetectorConstruction.hh"

//....oooOOOOOOooo.....oooOOOOOOooo.....oooOOOOOOooo.....oooOOOOOOooo.....
....

ExpPhysicsList::ExpPhysicsList()
: G4VUserPhysicsList()
{

  defaultCutValue = 0.00001*mm;

  cutForGamma = defaultCutValue;
  cutForElectron = defaultCutValue;

  cutForOpticalPhoton = defaultCutValue;

  VerboseLevel = 0;
  OpVerbLevel = 1;
SetVerboseLevel(VerboseLevel);
}

//....oooOOOOOOooo.....oooOOOOOOooo.....oooOOOOOOooo.....oooOOOOOOooo.....
....

ExpPhysicsList::~ExpPhysicsList()
{
//  delete physicsListMessenger;
}
```

```
//....oooOOOOOOooo........oooOOOOOOooo........oooOOOOOOooo........oooOOO
OOooo....

void ExpPhysicsList::ConstructParticle()
{
  // In this method, static member functions should be called
  // for all particles which you want to use.
  // This ensures that objects of these particle types will be
  // created in the program.

  ConstructBosons();
  ConstructLeptons();
  ConstructBarions();
  ConstructIons();
}

//....oooOOOOOOooo........oooOOOOOOooo........oooOOOOOOooo........oooOOO
OOooo....

void ExpPhysicsList::ConstructBosons()
{

  // gamma
  G4Gamma::GammaDefinition();

 //OpticalPhotons
  G4OpticalPhoton::OpticalPhotonDefinition();

}

//....oooOOOOOOooo........oooOOOOOOooo........oooOOOOOOooo........oooOOO
OOooo....

void ExpPhysicsList::ConstructLeptons()
{
  // leptons
  G4Electron::ElectronDefinition();
  G4Positron::PositronDefinition();
}
//....oooOOOOOOooo........oooOOOOOOooo........oooOOOOOOooo........oooOOO
OOooo....

void ExpPhysicsList::ConstructBarions()
{
  G4Proton::ProtonDefinition();
}
void ExpPhysicsList::ConstructIons()
{
//  Ions
 G4Alpha::AlphaDefinition();
}

void ExpPhysicsList::ConstructProcess()
{
  AddTransportation();
  ConstructEM();
  ConstructOp();
//  ConstructGeneral();
```

```
}

//....oooOOOOOOooo........oooOOOOOOooo........oooOOOOOOooo........oooOOO
OOooo....

#include "G4LowEnergyCompton.hh"
#include "G4LowEnergyGammaConversion.hh"
#include "G4LowEnergyPhotoElectric.hh"
#include "G4LowEnergyRayleigh.hh"

// e+
#include "G4MultipleScattering.hh"
#include "G4eIonisation.hh"
#include "G4eBremsstrahlung.hh"
#include "G4eplusAnnihilation.hh"

#include "G4LowEnergyIonisation.hh"
#include "G4LowEnergyBremsstrahlung.hh"
#include "G4hLowEnergyIonisation.hh"

//....oooOOOOOOooo........oooOOOOOOooo........oooOOOOOOooo........oooOOO
OOooo....

void ExpPhysicsList::ConstructEM()
{
  theParticleIterator->reset();
  while( (*theParticleIterator)() ){
    G4ParticleDefinition* particle = theParticleIterator->value();
    G4ProcessManager* pmanager = particle->GetProcessManager();
    G4String particleName = particle->GetParticleName();

    if (particleName == "gamma") {

      // gamma
      pmanager->AddDiscreteProcess(new G4LowEnergyCompton);

      LePeprocess = new G4LowEnergyPhotoElectric();
      pmanager->AddDiscreteProcess(LePeprocess);

      pmanager->AddDiscreteProcess(new G4LowEnergyRayleigh);

    } else if (particleName == "e-") {
      //electron
      pmanager->AddProcess(new G4MultipleScattering,-1, 1,1);

      LeIoprocess = new G4LowEnergyIonisation();
      pmanager->AddProcess(LeIoprocess, -1,  2, 2);

      LeBrprocess = new G4LowEnergyBremsstrahlung();
      pmanager->AddProcess(LeBrprocess, -1, -1, 3);

    } else if (particleName == "e+") {
      //positron
      pmanager->AddProcess(new G4MultipleScattering,-1, 1,1);
      pmanager->AddProcess(new G4eIonisation,      -1, 2,2);
      pmanager->AddProcess(new G4eBremsstrahlung,   -1,-1,3);
      pmanager->AddProcess(new G4eplusAnnihilation,  0,-1,4);
```

```
    }

   }

}

//....oooOOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOO
OOooo....

void ExpPhysicsList::SetGELowLimit(G4double lowcut)
{
  if (verboseLevel >0){
    G4cout << "ExpPhysicsList::SetCuts:";
    G4cout << "Gamma and Electron cut in energy: " << lowcut*MeV << " (MeV)"
<< G4endl;
  }

  G4Gamma::SetEnergyRange(lowcut,1e5);
  G4Electron::SetEnergyRange(lowcut,1e5);
  G4Positron::SetEnergyRange(lowcut,1e5);

}

void ExpPhysicsList::SetGammaLowLimit(G4double lowcut)
{
  if (verboseLevel >0){
    G4cout << "ExpPhysicsList::SetCuts:";
    G4cout << "Gamma cut in energy: " << lowcut*MeV << " (MeV)" << G4endl;
  }

  G4Gamma::SetEnergyRange(lowcut,1e5);

}

void ExpPhysicsList::SetElectronLowLimit(G4double lowcut)
{
  if (verboseLevel >0){

    G4cout << "ExpPhysicsList::SetCuts:";
    G4cout << "Electron cut in energy: " << lowcut*MeV << " (MeV)" << G4endl;

  }

  G4Electron::SetEnergyRange(lowcut,1e5);

}
void ExpPhysicsList::SetGammaCut(G4double val)
{
  ResetCuts();
  cutForGamma = val;
}

//....oooOOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOO
OOooo....

void ExpPhysicsList::SetElectronCut(G4double val)
{
```

```
  ResetCuts();
  cutForElectron = val;
}

void ExpPhysicsList::SetCuts(){

   SetCutValue(cutForGamma,"gamma");
   SetCutValue(cutForElectron,"e-");
   SetCutValue(cutForElectron,"e+");

   if (verboseLevel>0) DumpCutValuesTable();
}

//....oooOOOOOOooo.....oooOOOOOOooo.....oooOOOOOOooo.....oooOOO
OOooo....

void ExpPhysicsList::SetLowEnSecPhotCut(G4double cut){

  G4cout<<"Low energy secondary photons cut is now set to: "<<cut*MeV<<"
(MeV)"<<G4endl;
  G4cout<<"for processes LowEnergyBremsstrahlung,
LowEnergyPhotoElectric, LowEnergyIonisation"<<G4endl;
  LeBrprocess->SetCutForLowEnSecPhotons(cut);
  LePeprocess->SetCutForLowEnSecPhotons(cut);
  LeIoprocess->SetCutForLowEnSecPhotons(cut);
}

void ExpPhysicsList::SetLowEnSecElecCut(G4double cut){

  G4cout<<"Low energy secondary electrons cut is now set to: "<<cut*MeV<<"
(MeV)"<<G4endl;

  G4cout<<"for processes LowEnergyIonisation"<<G4endl;

  LeIoprocess->SetCutForLowEnSecElectrons(cut);
}
//....oooOOOOOOooo.....oooOOOOOOooo.....oooOOOOOOooo.....oooOOO
OOooo....
// Optical Processes
////////////////////////////////////////////////////////
#include "G4Scintillation.hh"
#include "G4OpAbsorption.hh"
#include "G4OpRayleigh.hh"
#include "G4OpBoundaryProcess.hh"

void ExpPhysicsList::ConstructOp() {

  // default scintillation process
  G4Scintillation* theScintProcessDef = new
G4Scintillation("Scintillation");
//  theScintProcessDef->DumpPhysicsTable();
  theScintProcessDef->SetTrackSecondariesFirst(true);
  theScintProcessDef->SetScintillationYieldFactor(1.0);
  theScintProcessDef->SetScintillationExcitationRatio(0.0);



//  theScintProcessDef->SetScintillationYield(8000./MeV);
```

```
//  theScintProcessDef->SetResolutionScale(0.);
//  theScintProcessDef->SetScintillationTime(2.7*ns);
  theScintProcessDef->SetVerboseLevel(OpVerbLevel);

// optical processes
  G4OpAbsorption* theAbsorptionProcess = new G4OpAbsorption();
  G4OpRayleigh* theRayleighScatteringProcess = new G4OpRayleigh();
  G4OpBoundaryProcess* theBoundaryProcess = new G4OpBoundaryProcess();
//  theAbsorptionProcess->DumpPhysicsTable();
//  theRayleighScatteringProcess->DumpPhysicsTable();
  theAbsorptionProcess->SetVerboseLevel(OpVerbLevel);
  theRayleighScatteringProcess->SetVerboseLevel(OpVerbLevel);
  theBoundaryProcess->SetVerboseLevel(OpVerbLevel);
  G4OpticalSurfaceModel themodel = unified;
  theBoundaryProcess->SetModel(themodel);

  theParticleIterator->reset();
  while( (*theParticleIterator)() ){
    G4ParticleDefinition* particle = theParticleIterator->value();
    G4ProcessManager* pmanager = particle->GetProcessManager();
    G4String particleName = particle->GetParticleName();
    if (theScintProcessDef->IsApplicable(*particle)) {

    pmanager->AddDiscreteProcess(theScintProcessDef);
    }
    if (particleName == "opticalphoton") {
      pmanager->AddDiscreteProcess(theAbsorptionProcess);
      pmanager->AddDiscreteProcess(theRayleighScatteringProcess);
      pmanager->AddDiscreteProcess(theBoundaryProcess);
  }
  }
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOO
OOooo....
```

PrimaryGeneratorAction.cc

```
// $Id: ExpPrimaryGeneratorAction.cc,v 1.10 2002/11/05 Ma Qing-li
// GEANT4 tag $Name: geant4-04-00 $

#include "ExpPrimaryGeneratorAction.hh"
#include "ExpDetectorConstruction.hh"
#include "G4Event.hh"
#include "G4ParticleGun.hh"
#include "G4ParticleTable.hh"
#include "G4ParticleDefinition.hh"
#include "G4ios.hh"
#include "G4UImanager.hh"
extern G4int j;
G4int beamangle;
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOO
OOooo....

ExpPrimaryGeneratorAction::ExpPrimaryGeneratorAction(
                                          ExpDetectorConstruction* myDC)
:myDetector(myDC)
```

```
{
  G4int n_particle = 5000;
  particleGun  = new G4ParticleGun(n_particle);


  // default particle kinematic

  G4ParticleTable* particleTable = G4ParticleTable::GetParticleTable();
  G4ParticleDefinition* particle =
particleTable->FindParticle("gamma");
  particleGun->SetParticleDefinition(particle);
  particleGun->SetParticleEnergy(1.0*MeV);
  particleGun->SetParticlePosition(G4ThreeVector(0,0,0));

}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOO
OOooo....

ExpPrimaryGeneratorAction::~ExpPrimaryGeneratorAction()
{
  delete particleGun;
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOO
OOooo....

void ExpPrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent)
{
  //this function is called at the begining of event



  G4double phib,tgb,xb, yb, zb;
  G4double FiberLength=myDetector->GetFiberLength();
  G4double SFD=myDetector->GetSFD();
  G4double FiberDiameter=myDetector->GetFiberDiameter();
  tgb=atan(FiberDiameter/SFD);

  beamangle=(anEvent->GetEventID());
  phib =(tgb)*(50-beamangle);

     xb =(SFD+(0.5*FiberLength))*sin(phib);
     yb = 0.0;
     zb = (SFD+(0.5*FiberLength));
//  G4cout<<"j="<<j<<G4endl;

  particleGun->SetParticleMomentumDirection(G4ThreeVector(xb,yb,zb));

  particleGun->GeneratePrimaryVertex(anEvent);


}

//////////////////////////////////////////////////////////////////////////
//
```

RunAction.cc

```
// $Id: ExpRunAction.cc,v 1.6 2002/11/05 Ma Qing-li
// GEANT4 tag $Name: geant4-04-00 $
//
//....oooOOOOOOOOO.....oooOOOOOOOOO.....oooOOOOOOOOO.....oooOOOOOOOOO.....
//....oooOOOOOOOOO.....oooOOOOOOOOO.....oooOOOOOOOOO.....oooOOOOOOOOO.....
#include "ExpDetectorConstruction.hh"
#include "ExpRunAction.hh"
#include "G4PVPlacement.hh"
#include "G4Run.hh"
#include "G4RunManager.hh"
#include "G4UImanager.hh"
#include "G4VVisManager.hh"
#include "G4ios.hh"
extern G4double Totalenergydepo[200][200];
extern G4int eventNO,runNO,scanNO;
extern G4int Totallight[200][200];
extern G4double theta;
G4int run;
G4double rotation[100];
//....oooOOOOOOOOO.....oooOOOOOOOOO.....oooOOOOOOOOO.....oooOOOOOOOOO.....

ExpRunAction::ExpRunAction( ExpDetectorConstruction* myDC)
:myDetector(myDC)
{}


ExpRunAction::~ExpRunAction()
{}

//....oooOOOOOOOOO.....oooOOOOOOOOO.....oooOOOOOOOOO.....oooOOOOOOOOO.....

void ExpRunAction::BeginOfRunAction(const G4Run* aRun)
{

G4cout << "### Run " << aRun->GetRunID() << " start." << G4endl;
 run=aRun->GetRunID();
 G4double theta=myDetector->Gettheta();
rotation[run]=theta;

  if (G4VVisManager::GetConcreteInstance())
    {
      G4UImanager* UI = G4UImanager::GetUIpointer();
      UI->ApplyCommand("/vis/scene/notifyHandlers");
      UI->ApplyCommand("/vis/viewer/refresh");

    }
}

//....oooOOOOOOOOO.....oooOOOOOOOOO.....oooOOOOOOOOO.....oooOOOOOOOOO.....

void ExpRunAction::EndOfRunAction(const G4Run*)
```

```
{
  if (G4VVisManager::GetConcreteInstance())
    {
     G4UImanager::GetUIpointer()->ApplyCommand("/vis/viewer/refresh");
    }


  if (run==scanNO)
    {
    for (runNO=0;runNO<=scanNO;runNO++)
       {
        G4cout<<"["<<rotation[runNO]<<"]"<<G4endl;

       for (eventNO=0;eventNO<=99;eventNO++)
          {
           G4cout<<eventNO<<"
"<<Totalenergydepo[eventNO][runNO]<<G4endl;
          }
        }
    for (runNO=0;runNO<=scanNO;runNO++)
      {
       G4cout<<"["<<rotation[runNO]<<"]"<<G4endl;
       for (eventNO=0;eventNO<=99;eventNO++)
          {
           G4cout<<eventNO<<" "<<Totallight[eventNO][runNO]<<G4endl;
          }
       }
     }
}
//....oooOOOOOOooo........oooOOOOOOooo........oooOOOOOOooo........oooOOO
OOooo......
```

EventAction.cc

```
// $Id: ExpEventAction.cc,v 1.7 2002/11/05 Ma Qing-li
// GEANT4 tag $Name: geant4-04-00 $
//
//....oooOOOOOOooo........oooOOOOOOooo........oooOOOOOOooo........oooOOO
OOooo......
//....oooOOOOOOooo........oooOOOOOOooo........oooOOOOOOooo........oooOOO
OOooo......

#include "ExpEventAction.hh"

#include "G4Event.hh"
#include "G4EventManager.hh"
#include "G4TrajectoryContainer.hh"
#include "G4Trajectory.hh"
#include "G4VVisManager.hh"
#include "G4ios.hh"

//....oooOOOOOOooo........oooOOOOOOooo........oooOOOOOOooo........oooOOO
OOooo......

 G4int eventNO;

ExpEventAction::ExpEventAction()
```

```
{}

//....oooOOOOOOooo........oooOOOOOOooo........oooOOOOOOooo........oooOOO
OOooo......

ExpEventAction::~ExpEventAction()
{}

//....oooOOOOOOooo........oooOOOOOOooo........oooOOOOOOooo........oooOOO
OOooo......

void ExpEventAction::BeginOfEventAction(const G4Event*)
{}

//....oooOOOOOOooo........oooOOOOOOooo........oooOOOOOOooo........oooOOO
OOooo......

void ExpEventAction::EndOfEventAction(const G4Event* evt)
{
 G4int event_id = evt->GetEventID();
 eventNO=evt->GetEventID();


  // get number of stored trajectories
  G4TrajectoryContainer* trajectoryContainer =
evt->GetTrajectoryContainer();
  G4int n_trajectories = 0;

  if (trajectoryContainer) n_trajectories =
trajectoryContainer->entries();

  // periodic printing

  if (event_id < 100000 || event_id%100000 == 0) {
//    G4cout << ">>> Event " << evt->GetEventID() << G4endl;

//    G4cout << "    " << n_trajectories
//     << " trajectories stored in this event." << G4endl;
  }

//
// extract the trajectories and draw them
  //
  if (G4VVisManager::GetConcreteInstance())
    {
     for (G4int i=0; i<n_trajectories; i++)
        { G4Trajectory* trj = (G4Trajectory*)
                              ((*(evt->GetTrajectoryContainer()))[i]);
          trj->DrawTrajectory(50);

        }
    }
}

//....oooOOOOOOooo........oooOOOOOOooo........oooOOOOOOooo........oooOOO
OOooo......
```

攻读学位期间主要研究工作和发表论文情况：

1. 马庆力，詹志锋等 “NSRL 中的 BPM 前端电子学系统研究”,核电子学于探测技术 Vol.23 No.4 2003 p: 343-346
2. 马庆力，詹志锋等，“NSRL 的 BPM 系统中对数比电路的设计与研制”，电子测量与仪器学报 Vol.17 No.4 2003 p: 15-19
3. 马庆力等，“闪烁光纤在 r 射线辐照下部分特性的蒙特卡罗模拟”，强激光与粒子束 Vol.16 No.3 2004 p: 385-388
4. 马庆力等，“运用 GEANT4 模拟闪烁光纤阵列特性与光纤长度及入射能量的关系”，核技术 Vol.28 No.2 2005
5. 马庆力，詹志锋等，“对数比电路在加速器束团位置监测中的应用与测量”，核技术，Vol.27 No.2 2004 p:91-95
6. 马庆力，M.M.Nasseri 等，“闪烁光纤阵列用于高能射线成像的可行性研究”，核技术（accepted）
7. 詹志锋，马庆力等，“合肥光源逐圈束流位置监测系统中的定时系统”，强激光与粒子束，Vol.15 No.5 2003 p: 517-520
8. 居桐，肖延国，赵军平，马庆力，詹志锋等，“基于 PCI 总线直接 MASTER 模式的高速数据采集”，核电子学于探测技术，Vol.22 No.5 2002 p:456-458
9. M M . Nasseri, **Ma Qingli** et.al. “Monte-Carlo Simulation for Determining SNR and DQE of Linear Array Plastic Scintillating Fiber”, Nuclear Science & Technique, Vol.15 No.5 Oct.2004.p:304-307
10. M M . Nasseri, **Ma Qingli** et.al. “Monte-Carlo Simulation to Determine Detector Efficiency of Plastic Scintillating Fiber”, Nuclear Science & Technique, Vol.15 No.5 Oct.2004. p:308-311
11. M M . Nasseri, **Ma Qingli** et.al. “Image quality evaluation of linear plastic scintillating fiber array detector for X-ray imaging”, Nuclear Science & Technique Vol.15 No.6 Oct.2004 p:361-364
12. M.M.Nasseri, **Qingli Ma**, Zejie Yin et.al. "Low Energy X-ray Imaging Using Plastic Scintillating Fiber: a Simulation Study", Nuclear Instruments and Methods in Physics Research B, Available online 16 March 2005