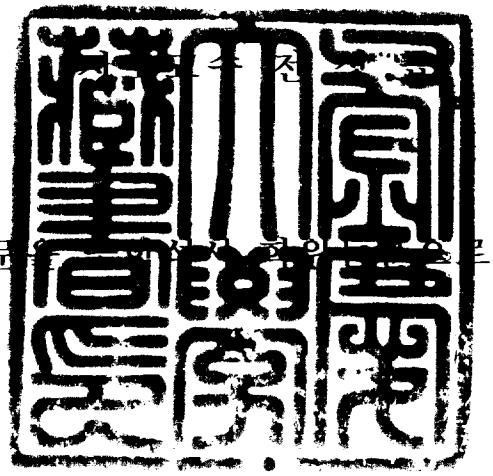
공 학 석 사 학 위 논 문

Computerized Tomography를 이용한 골밀도 진단기의 설계



이 논문을 공학석사 학위논문으로 제출함

2005년 2월

부 경 대 학 교 대 학 원

전자공학과

허 준

허준의 공학석사 학위논문을 인준함

2004년 12월 22일

주 심 공학박사 허 만 탁 

위 원 공학박사 김 종 진 

위 원 공학박사 전 성 즙 

목 차

Bone Density Measurement System Adopting Computerized Tomography

Joon Heo

Department of Electronics Engineering,Graduate School.
Pukyong National University.

Abstract

Recently, adult disease is increasing rapidly as the dietary life is westernized. Especially entering an aging society, osteoporosis is increasing rapidly.

Osteoporosis is a silent progressive disease that results in fracture in nearly 1 out of 2 women and 1 out of 6 men. The majority of these fractures occur in post-menopausal women, secondary osteoporosis affects men and women of all ages.

There are two kind of method for measurement of osteoporosis. One is a method using ultra sonic transducer, the other is a method using X-ray. The method using X-ray is got more accurate data than the other one. However, it is not easy to get an anatomical position or mutual relationship among measured points from the X-ray image obtained from the existing medical appliances.

In this paper, a new bone densitometer adopting the principle of the CT(Computerized Tomography) is designed. The designed bone densitometer acquires 270 plane images rotating around the bone at one degree interval, from which Voxel values are reconstructed. Voxels are three-dimensional and contain the values of bone density.

In the designed bone densitometer, X-rays with two different energy levels are shot two times to obtain one plane image. Offset values given by trabecular bone, cortical bone and tissue are eliminated. According, more accurate image that could not get with the existing bone densitometers are extracted.

제 1 장 서 론

최근 의료 선진국 일수록 국민 보건에 대한 관심과 질병을 조기에 정확히 진단할 수 있는 첨단 의료 기기의 역할이 더욱 더 증대 되고 있다. 특히, 암이나 치매와 같은 각종 질병에 대한 의료 서비스의 기술이 고도화 될수록 의사들의 의료 기기에 대한 의존도는 더욱 높아질 수밖에 없다. 일반적으로 의료 서비스의 질은 의료 기기에 의해 좌우된다고 해도 과언이 아니다. 따라서 의료 기기 산업은 뉴밀레니엄 시대에서 정보 기술에 이어 차기 경제 성장의 이슈로 거론되고 있다. 아직 국내 의료 기기 산업은 걸음마 단계에 불과하지만 정부에서는 2010년까지 전자 의료기기 산업을 선진국 수준에 진입시킨다는 목표를 가지고 있다.

최근 서구화된 식생활 문화 등으로 인한 성인병이 급증하고 있는 추세에 있다. 특히, 고령화 사회로 접어들면서 폐경기 이후 여성들과 50대 이후 남성들 사이에서 골다공증이 급속히 증가하고 있다. 일반적으로 이러한 골다공증을 진단하는 방식으로는 초음파 방식과 X-선 방식을 주로 사용하고 있다. X-선 방식의 경우 초음파 방식보다 정확한 데이터를 제공하기 때문에 더욱 많이 사용되고 있다. [3]

기존의 X-ray Film에 투영된 평면 이미지로는 해부학적인 위치나 상호관계를 진단하기 어렵다는 단점을 가지고 있다. 이러한 단점을 보완하기 위해 컴퓨터 단층 촬영(Computerized Tomography :CT)기가 개발되었다.[3][4] 그리고, 골다공증 진단시 CT장비를 이용하여 골밀도값을 측정 하기도 한다. 이때 CT장비 구동시 에너지·시간·비용부담이 상당히 크다는 단점이 있다.

이러한 점에 착안하여 본 논문에서는 골밀도 진단에 있어서 전신용 CT장비의 단점이 보완된 Forearm과 Heel만으로 짧은시간과 저비용으로 골밀도를 진단할 수 있는 기기를 설계하였다. 기존의 골밀도 진단기기들은 평면의 이미지를 분석하여 골밀도값을 얻기 때문에 소주골(Trabecular Bone)과 피질골(Cotical Bone) 그리고 Tissue등 모든성분들이 골밀도 데이터에 포함될 수 밖에없다.

본 논문에서 CT방식을 적용하여 얻은 단면 이미지의 골밀도 데이터는 기존 골밀도 진단기기들에 포함되어 있는 피질골(Cotical Bone)[2]과 Tissue성분을 완전히 제거 하였고, 순수한 소주골(Trabecular Bone)[2]만의 데이터로 더욱 정확한 골밀도 측정값을 얻었다.

제 2 장 골밀도 진단기의 이론 및 적용

2.1 초음파 방식

초음파 방식은 두개의 초음파센서(Ultra Sonic Transduser:송·수신겸용)[4]를 이용하여 골밀도를 측정하는 방법이다. 초음파 방식의 경우에는 송신부의 변환기에서 초음파를 측정부위에 발생시킬 때, 매질에 대한 음파의 굴절·반사율이 상대적으로 크고 , 투과율은 적은 반면에 수신부의 변환기에서 측정된 초음파의 속도, 대역폭 그리고 크기가 동일 매질이라도 매질의 미세 위치의 오차와 각도에 따른 오차율이 크다는 단점이 있다. 이는 초음파센서의 극세화로 많은 수의 초음파센서를 어레이로 적용을 하여, 데이터의 오차를 최대한 줄이고, 밀도측정부위를 이미지화 하여 3D 기술을 이용하면 좋은 결과를 얻을 수 있다.

2.2 Dual X-ray 방식

Dual x-ray source를 이용하여 골밀도 값을 측정하는 방식은 초음파센서를 이용한 방식보다는 매우 정확한 골밀도 값을 측정할 수 있는 장점을 가지고 있다. 하지만 단지 이차원적 평면의 데이터로 밀도를 측정할 수밖에 없다는 단점이 있다. 그리고 Dual X-ray 방식을 사용하면, 초음파 고유의 특성에 의한 에너지의 손실을 최대로 줄일 수 있다. 만약 두 에너지 중 Low 에너지가 70kV이고, High 에너지가 140kV라고 가정을 하면 골밀도 값은 간단히 아래 수식을 적용하여 얻을 수 있다.[1][2][3]

$$BMD=BMC/Area$$

$$BMC=KH-L$$

$$K=(Htissue-Hair)-(Ltissue-Lair)$$

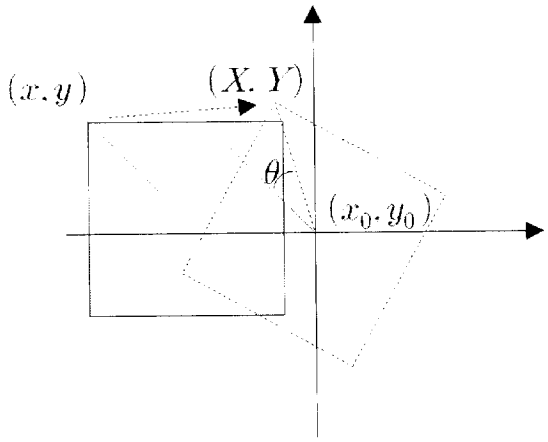
BMD : bone mineral density

BMC : bone mineral content (2-1)

2.3 CT (Computerized Tomography) 방식

본 논문에서 설계한 장치는 듀얼 에너지 (high : 60kV, 0.8mA / Low : 50KV, 0.8mA)의 X-선을 피사체에 투과하여 평면의 데이터를 얻고, 1도 간격으로 얻어진 데이터를 270도 까지 수집하여 단면의 이미지를 얻게 된다. 단면 이미지를 얻은 후에 확대 정수 값에 의한 CT Number를 얻고, CT의 voxel 값으로 치환한 다음 밀도 값을 얻을 수 있다. 이때, 디텍터에서 감지된 영상의 단면을 재구성하여 수많은 작은 Block 으로 표현한 것이 CT영상으로 조합된다. 이러한 작은 Block을 voxel이라고 한다. 이때 voxel은 X-선의 감약과 비례한 값을 갖는다. voxel의 두께(x)는 측정 가능한 값이므로 한 voxel의 선감약계수(μ)는 지수감약공식을 이용하여 쉽게 구할 수 있다.[3][4][5]

제작된 골밀도 진단기에서는 방향(1도)이 다른 270개의 추가정보를 가지고 있으므로 아래의 회전공식과 방정식을 이용하여 각 voxel 선감약계수를 구할 수 있다.



[그림 2.1] 골밀도 진단기의 회전 변환

$$X=(x-x_0)\cos\theta+(y-y_0)\sin\theta+x_0$$
$$X=-(x-x_0)\sin\theta+(y-y_0)\cos\theta+y_0 \qquad (2\text{-}2)$$

회전후의 좌표는 좌표(X, Y), 회전하기전의 좌표는 좌표(x, y) 그리고 상대적 중점 좌표는 좌표(x_0,y_0) 이다.

$$\begin{aligned}
N_1 &= N_0 e^{-(\mu 1+\mu 2+\mu 3+\ldots+\mu 639+\mu 640)x}\\
N_2 &= N_0 e^{-(\mu 1-\mu 2+\mu 3+\ldots+\mu 639-\mu 640)x}\\
N_3 &= N_0 e^{-(\mu 1+\mu 2+\mu 3-\ldots-\mu 639+\mu 640)x}\\
&\vdots\\
N_{269} &= N_0 e^{-(\mu 1+\mu 2+\mu 3+\ldots+\mu 639+\mu 640)x}\\
N_{270} &= N_0 e^{-(\mu 1+\mu 2+\mu 3+\ldots+\mu 639+\mu 640)x}
\end{aligned} \qquad (2\text{-}3)$$

(N_0:입자광자의수, μ:선감약계수)

CT에서는 물의 선감약계수를 표준으로 하여 뼈의 선감약계수를

1000, 공기의 선감약계수를 -1000의 확대 정수 값을 구하여 CT 번호를 부여하며 유도식은 아래와 같다.[4]

$$CT_{Number} = K\frac{\mu_p - \mu_w}{\mu_w} \qquad (2\text{-}4)$$

K : 확대정수

μ_p : 알지 못하는 voxel의 선감약계수

μ_w : 물의 선감약계수

최종 재구성된 두개의 단면 이미지를 이용한 골밀도 값은 간단히 식(2-5)를 적용하여 얻을 수 있다.

BMD=BMC/Area

BMC=KH-L

K=(Htisnum-Hairnum)-(Ltisnum-Lairnum)

BMD : bone mineral density

BMC : bone mineral content (2-5)

tisnum : tissue CT Number

airnum : air CT Number

제 3 장 골밀도 진단기 측정 이론

3.1 Televised Electronic Tomography

Televised Electronic Tomography는 필름 cassette를 이용하는 종래의 단층 검사와는 달리 필름을 영상증배관(Image Intensifier)으로 대체한 다음, X-선관과 함께 동기시켜 이동하도록 하는 방식이다. 한 단층면을 검사하는 동안 카메라를 통하여 얻어지는 영상 데이터들을 video tape recorder 혹은 video disk storage에 기록하게 된다. 이렇게 얻어진 영상 데이터들을 공간중복을 할 수 있는 영상기억장치(storage tube)를 이용하여 영상데이터들을 서로 겹쳐서 단층영상을 만드는 방법이다.

기존의 단층촬영 기법은 한 장의 필름에 X-선 노출을 하기 때문에 영상에 나타난 구조물은 X-선관이 이동하는 동안 연속 발생하는 영상들이 한 장의 필름에 모두 겹쳐서 나타나게 된다. 이러한 방법은 X-선관이 이동하면서 연속적으로 발생하는 영상들을 각기 따로 녹화하였다가 나중에 겹쳐 단층영상을 만들어 낸다는 점에서 차이가 있다.

3.2 확대 촬영술

일반적으로 영상 이미지의 확대는 렌즈 또는 확대경 등을 이용한 광학적 확대(Optical Magnification)와 피사체와 필름간의 거리를 조절하여 직접적으로 X-선을 이용하여 확대하는 방사선 확대촬영(Radiographic Magnification)으로 크게 나눌 수 있다. 본 논문에서는 방사선 확대 촬영술을 응용하였다.

3.2.1 진확대와 기하학적 확대

실제 X-선관의 초점은 아무리 작다고 할지라도 어느 정도의 물리적 면적을 갖게 마련이다. 그러나 X-선 발생원이 point source 라고 가정한다면 진확대(M)와 기하학적 확대(m)는 정확하게 같아진다. 하지만 불행하게도 X-선 발생원은 점일 수가 없기 때문에 일정한 면적을 갖게 마련이다. 이로 인해 실제로 확대된 영상의 크기는 중앙부의 진음영(umbra)과 주위의 가음영(penumbra)을 합한 크기가 된다. 결과적으로 초점의 크기로 인하여 진확대와 기하학적 확대는 달라지고 그 관계는 다음과 같은 식으로 표현된다.

$$\mathbf{M = m + (m-1)\ (\frac{f}{d})} \qquad (3\text{-}1)$$

M = 진확대

m = 기하학적 확대

d = 피사체의 크기

f = 초점의 크기

3.2.2 X-선관 초점

확대촬영을 하기 위해서는 작은 초점을 사용하여 가능한 기하학적 불선예도를 감소시켜야 하므로 보통 X-선관의 초점 크기가 0.3mm 또는 그 이하인 것을 주로 사용한다.

(1) 초점의 크기

가) 실효초점

실효초점이란 일반적으로 방출 X-선의 중심선속(central beam)방향에서 관측한 초점의 크기를 말한다.

나) 등가초점

등가초점은 실효초점의 일종으로 pinhole camera법으로 측정한 크기(실효초점)가 동일한 초점일지라도 에너지 강도 분포상태에 따라서 해상도가 달라진다.

(2) 초점의 측정

초점의 측정방법은 초점의 크기에 따라 적당한 측정법을 선택해야한다. ICRU에서는 0.3mm 또는 그 이하 크기의 초점은 해상능측정법(line-pair resolution test method)을 사용하도록 하고, 0.3mm 이상의 초점의 경우에는 pinhole camera법을 이용하도록 권고하고 있다.

가) Pinhole camera 측정법

Pinhole camera 측정법은 초점의 물리적 크기를 측정하는 방법으로 납판에 작은 구멍(pinhole)을 뚫어 X-선관과 필름사이에 놓고 필름의 농도가 0.6~1.0이 되도록 X-선을 노출하여 초점의 영상을 얻는 방법이다. 이 방법은 초점의 물리적 크기를 측정 할 수 있을 뿐 아니라 초점에서 방출되는 방사선의 에너지 강도분포(energy intensity distribution)도 알 수 있다. Pinhole법으로 측정한 초점의 크기(실효초점)는 확대촬영에서 사진확대를 계산할 때의 초점크기와 같고, 에너지 강도분포로는 양극축열 특성을 산출할 수 있다. 강도분포가 다른 초점일지라도 그 크기의 측정값은 동일하다

(3) 노출조건(mA, kVp)에 따른 초점크기의 변화

초점의 크기는 노출조건에 따라 변화하는데 관전류(mA)에 비례하여 증가한다. 이것은 높은 관전류에서 초점이 더욱 작열(blooming)하기 때문이다. 그렇기 때문에 낮은 관전압 그리고 높은 관전류 일수록 증가율은 증가하게 된다. 노출조건변화에 따른 초점크기의 변화는 [표 4.1]과 같다.

[표 4.1] 노출조건변화에 따른 초점 크기의 변화

관전압(kVp)	관전류(mA)	초점의 크기
40 kVp	100mA	2.0mm
40	300	2.3mm
80	100	1.7mm
80	300	1.8mm

3.2.3 분해능 평가

확대촬영술은 적정한 필름과 증감지의 선정, 정확한 초점의 크기 그리고 확대율 등이 적당히 조화를 이루어야만 최대의 효과를 얻을 수 있다. 바로 이들 각 구성인자들의 MTF(Modulation Transfer Function) 합성값으로 총 영상계의 능력을 평가하기 때문에 방사선 영상계의 구성인자들의 해상도 특성을 알아야 한다. 만약 MTF 값이 1이라면 완전한 상태로서 피사체가 포함하고 있는 모든 정보를 하나도 빠짐없이 기록할 수 있다는 것을 의미한다. 반대로 MTF 값이 0이라는 것은 피사체의 정보를 하나도 기록할 수 없다는 것을 의미하게 된다. 확대촬영시 영상의 구성인자중 X-선 장치를 제외하면 필름, 초점, 증감지, 피사체의 움직임 등이 기록계의 중요인자들이므로 이들의 MTF가 확대촬영에 어떠한 영향을 끼치는지 알아야할 필요가 있다.

(1) 필름

10~20 Line/mm의 주파수를 지닌 피사체를 기록할 수 있는 필름이라면 상당한 능력을 가진 것으로 확대촬영시 MTF 값이 1이라고 취급해도 무방하다.

(2) 초점

일정초점크기에서는 확대율을 크게 할수록 MTF 값이 떨어지며, 초점이 클수록 확대율을 증가시키며 초점이 작을 때 보다 MTF 값이 급격히 감소하게 된다.

(3) 증감지

일반확대촬영술 혹은 혈관확대촬영술의 경우에는 노출조건을 높여 주어야 하기 때문에 노출시간을 단축시키기 위해 고감도의 증감지를 사용하는 것을 원칙으로 한다. 일반적으로 증감지의 MTF 값은 초점의 확대율과 함께 증가하게 된다.

(4) 초점과 증감지의 관계

확대를 할수록 초점의 MTF 값은 감소하고 증감지의 MTF 값은 증가한다. 따라서 적정한 초점을 사용하여 움직이는 혈관을 확대촬영 할 때, 고감도 증감지를 사용하여 가능한 노출시간을 짧게 하면 움직임에 의한 불선예도를 최대한으로 억제할 수 있기 때문에 혈관 조영술의 전체 MTF 값은 는 향상된다.

확대촬영의 해상도을 이해하려면 영상체계의 모든 구성요소들의 MTF를 관찰하여야한다. 만약 필름의 MTF를 1.0 이라고 했을 때 증감지의 MTF는 확대와 함께 증가하지만 초점의 MTF는 확대에 따라 급격히 감소한다. 확대촬영의 재현성은 초점과 증감지를 적절히 결합할 때 최대를 이루게 된다. 확대란 고감도 증감지를 이용하고 피사체의 움직임이 예상될 때 효과를 얻을 수 있다. 피사체 움직임에 의한 불선예도의 정도는 확대율과는 무관하고 움직임의 속도 노출시간과 관련되어 있다. 적정한 확대 촬영은 0.3mm 초점일 때 확대율 1.6 ~ 2일 때이다.

3.3 Subtraction 방법

Subtraction방법은 사진영상에서 원하지 않는 음영을 삭제하기 위한 사진학적인 방법이다. 하지만 사진학적 기술로 어떠한 정보를 추가 기록하는 것이 아니라 진단적으로 필요한 부분을 쉽게 관찰할 수 있도록 하는 방법의 일종이다. 방사선진단에서 Subtraction방법은 1935년 독일의 Ziedes des Plantes가 처음으로 사용하였다. Subtraction방법을 혈관조영술에 응용하여 연속필름에 조영제가 주입된 혈관만을 쉽게 관찰할 수 있다.

뇌혈관조영검사에서 subtraction방법을 이용하려면 다음 세가지 조건이 충족되어야 한다. 첫째, 단순사진(scout film)일 것, 둘째, 혈관에 조영제가 주입된 혈관조영상일 것 그리고 뇌혈관조영 검사시 환자의 움직임이 없어야 한다는 것이다.

3.4 전산화 단층 촬영술

전산화 단층 촬영술(Computerized Axial Transverse Scanning)은 1972년 영국의 EMI 수석연구관인 GN. Hounsfield가 개발한 새로운 영상기술이다. 이것은 머리의 얇은 횡단면(tomographic slice)을 연필형 X-선속(pencil-beam)을 여러 방향으로 조사하여 투과 X-선을 형광검출기로 수집하게 된다. 그렇게 수집된 결과를 컴퓨터에서 감약정도를 수학적 산술법(mathematical algorithm)으로 분석하고 다시 단층 영상을 재구성(reconstructed)하는 방법이다. 이렇게 만들어진 단층 영상은 일반 X-선 영상에서는 일찍이 볼 수 없었던 연부조직(혈액, 뇌척수액, 회질, 백질, 종양 등)과 같은 작은 부분의 영상도 얻을 수 있었다.

3.4.1 데이터 수집 (Data Accumulation)

(1) Scanning motions

CT의 세대는 인간사회의 세대와 같이 정의하기는 힘들지만 그 표준을 데이터를 얻는데 필요한 운동방식으로 구분하고 있다. 일반적으로 1977년까지는 그 발전단계에 따라 다음의 4개의 세대로 구분하고 있다.

가) 제1세대

제 1세대는 한 쌍 또는 1개의 검출기를 사용하였으며 연필형 X-선속을 이용하여 직선운동과 회전운동을 반복하는 구조이다. 이러한 형태로는 초기의 EMI 장치는 제 1세대에 해당한다.

나) 제2세대

제2세대 스캐너의 주 장점은 주사시간의 단축이다. 주사시간이 짧아진 만큼 환자의 부담도 많이 감소하고, 장치의 효능도 상당히 향상되었다. 만일 CT장치의 주사시간이 매우 빠르다면 전신주사시 환자에 의한 영상의 부정확성을 최소로 할 수 있다. 운동방식은 제 1세대 스캐너와

마찬가지로 직선운동과 회전운동을 이용하지만, 제1세대장치가 직선운동이 끝나면 1^0씩 회전하는데 비하여 제2세대 장치는 회전각도가 약 30^0 정도로 매우 크다. 반복되는 직선운동의 횟수는 검출기배열장치 내의 검출기 수량에 의해 결정된다. 즉 주사당 직선운동의 수는 검출기가 많아지면 작아지게 된다.

다) 제3세대

제3세대 스캐너는 제2세대 스캐너보다 더욱 많은 검출기를 이용하며, 부채꼴선속의 각도 대폭 증가하였다. 부채꼴선속은 환자 전체를 포함할 수 있을 정도로 각도를 크게 하였고 검출기의 수는 약 300개이다. 결론적으로 제3세대 스캐너에는 직선운동이 없는 것이 특징이다. 1회전운동이 끝나면 하나의 단층영상을 형성하며, 모든 스캐너가 그러하듯이 이 장치도 투과방사선을 측정키 위한 정지운동은 없고 움직임에 대한 보정은 컴퓨터 프로그램이 당당하게 된다.

라) 제4세대

제4세대 스캐너의 운동방식은 환자를 중심으로 원형으로 검출기를 고정시키고 X-선관만 이 360^0 회전운동을 하는 구조로 되어 있다. 4세대 스캐너의 주사시간은 3세대보다 빠르기는 하지만 그렇다고 대폭적 감소는 성취하지 못하였다. 4세대의 주된 장점은 검출기의 보정이 손쉽다는 것이다.

(2) X-선관

CT에 이용하는 X-선발생원이란 단백방상선을 공급하는 것이라야 영상의 재구성이 간단하고 정확도가 향상된다. CT용 X-선관이라고 해서 진단용 X-선관과 크게 다른 점은 없지만, 진단용 X-선관보다는 CT용 X-선관에서의 발열량이 높기 때문에 CT용 회전양극관의 경우 저자극 원반의 가장자리를 달리 하였다. CT용 X-선관의 양극원반은 특수한 형태로 하여 열충격과 원심력의 영향을 최소화 하였으며, 원반의 비틀림(warp)과 패임(crack)을 방지하였다.

(3) Collimator

X-선속의 조준은 X-선관측과 검출기측에서 반드시 행하며 대칭적으로 정확히 일치하도록 배열 조준한다. X-선관측 선속의 조준은 X-선관의 heel effect의 영향을 극소화할 수 있도록 조절방향 선택을 신중히 해야 한다. 검출기의 조준장치 사용목적은 산란선방지와 단편두께(voxel 의 길이)를 결정하는데 있다.

(4) 검출기 (Detector)

선속에너지를 측정하는 것은 CT검출기이다. 선속에너지는 각 광자의 에너지와 그 양에 의해 결정되는 것이다. 검출기는 방사성핵을 검출하는 scintillator와 같은 원리를 이용한다. 방사성핵 검출기는 좁은 범위의 에너지를 측정하도록 하여 이 에너지보다 더 높거나 낮은 에너지는 무시한다. Gating은 모든 입사광자에너지가 동일할 때(단색방사선) 주 광자에네지보다 낮은 산란 광자를 식별 분리하는데 가장 좋은 방법이다.

3.4.2 영상의 재구성

인체의 단편을 수많은 벽돌모양으로 나누어 표현한 것이 CT영상이다. 이러한 벽돌을 voxel(volume element)이라 하며, voxel은 X-선속이 벽돌에 의한 감약과 비례한다. voxel의 감약정도는 그 구성물과 두께 그리고 선속의 질에 따라 결정되며 이를 선감약계수(liner attenuation coefficient)라 한다. 수학적 풀이를 하면 우선 균질조직 한토막(벽돌)을 단색 X-선으로 조사하였다고 가정할 때 토막에 의한 선속의 감약정도 즉 선감약계수는 지수감약공식을 응용하여 풀이할 수 있다.

(1) 영상재구성의 연산방법

제공된 문제를 풀기 위한 수학적 방법을 알고리즘(algorithm)이라고 한다. 영상 matrix 내선감약계수를 수천개의 방정식을 이용해 풀어야만 한

다. Brooks와 DeChiro등은 1975년 연산 재구성 연산 방법을 Back-projection(중첩재구성법), Iterative methods (반복재구성법) 그리고 Analytical methods (분석적 재구성법)로 분류하였다.

가) 중첩재구성법

X-선속을 이동하여 물체를 상단과 좌측방향으로 직선 주사하면 계단 모양의 투영도가 형성된다. 각 계단 높이는 물체를 통과한 방사선량에 비례하기 때문에 중심부에서는 많은 양의 방사선이 투과할 수 있어 계단의 높이는 제일 높게 나타난다. 이 계단을 그들 높이에 맞추어 gray scale density를 부여하면 또 하나의 투영도가 형성된다. 이렇게 두 방향에서 얻어진 투영도를 back-projected 즉, 겹쳐 놓으면 원래 물체의 대략적인 윤곽을 나타내는 재구성영상이 형성된다. 만일 여러 방향에서 얻은 측면의 투영도를 더하면 이러한 영상의 품질은 상승하게 된다.

나) 반복재구성법

반복재구성법(Iterative Method)은 matrix 내의 모든 점이 같은 값을 지니고 있다고 가정하고 이 값과 측정값이 같거나 만족스러울 때까지 반복하여 보정작업을 하는 것이다. 반복재구성법은 세 가지 유형이 있다.

① 동시 재구성 (simultaneous reconstruction)

전체 matrix를 위한 모든 투영 값은 반복의 첫 시작부터 계산되고 모든 보정이 각 반복과정마다 동시에 이루어진다.

② 열 대 열 보정 (ray by ray correction)

한 열의 합을 계산, 보정하고 이들 보정은 다음 열과 연결되어 각 반복의 배열을 위해반복 처리 된다.

③ 점 대 점 보정 (point by point correction)

한 점을 투과한 모든 투영에 계산과 보정을 한다. 이들 보정은 연속 계산하여 합성하고 다시 매점에서 반복 처리하는 방법이다.

다) 분석적 재구성법

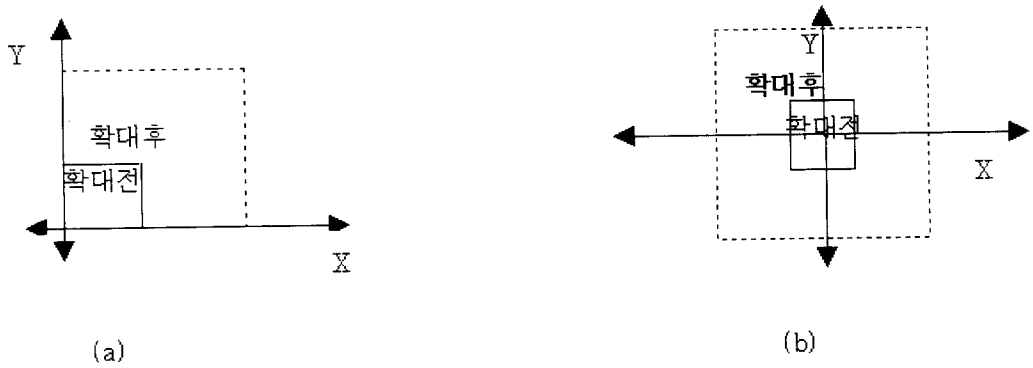
반복재구성법과는 달리 분석학적으로 영상을 재구성하는 방법으로 2차원적 Fourier 분석법과 여과중첩재구성법(filtered back-projection)이 있다.

제 4 장 영상 데이터의 기하학적 처리

4.1 기하학적 변환

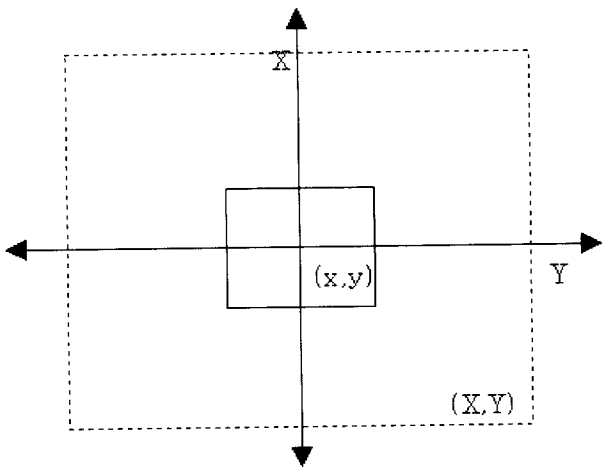
형태를 변환시키는 변환을 영상 처리의 세계에서는 기하학적 변환이라고 부른다. 기하학적 변환은 여러 분야에서 이용되고 있는데, 예를 들면, 기상 예보에서 기상위성이 촬영한 구름의 영상도 기하학 변환을 한 영상이라 할 수 있다. 인공위성에서 카메라로 촬영한 영상은 렌즈 등에 의한 왜곡을 포함하기 때문이다. 이러한 영상에 대하여 기하학적 변환을 가하여 보정을 하여 왜곡이 없는 영상이 되는 것이다. 카메라로서 파노라마 풍의 여러 장의 사진을 찍어 합성하는 부분은 어려운 문제이다. 연결부분이 불연속이라든지, 직선이 구부러지든지 하기 때문에, 훌륭히 합성하도록 하기 위해서는 확대를 한다든지 위치를 옮긴다든지 하지 않으면 안 된다. 따라서 기하학적 변환이란 화소의 위치의 변화 즉, 좌표평면에서 원 영상의 좌표변환을 통하여 새로운 형태의 결과 영상을 얻는 것을 말한다.

기하학적 변환에는 확대, 축소, 이동 등을 포함하여 각종 처리 기법이 있는데 단순한 처리 기법부터 살펴보도록 한다. 보통 영상 처리의 좌표 표현은 [그림 4.1]의 (a)에서 볼 수 있는 바와 같이, 좌 상단을 원점으로 해서, 오른쪽 방향과 아래 방향으로 좌표 값이 커지는 좌표계를 사용하고 있지만, 이 좌표계의 원점을 중심으로 확대하면, 그림(b)처럼 영상의 정 중앙을 원점으로 정하여 처리하는 쪽이 편리하다. [5][9]



[그림 4.1] 기학학적 변환

4.2 크기변환



[그림 4.2] 크기변환

어떤 점 (x,y)가 확대, 축소되어 (X,Y)로 위치가 변하면, 두 좌표 사이에는

$$\mathbf{X} = \mathbf{ax}$$
$$\mathbf{Y} = \mathbf{by} \qquad (4\text{-}1)$$

의 관계가 성립한다. 식 (4-1)에서 a, b는 각각 x방향, y방향의 확대율을 의미

한다. 식 (4-1)를 이용하여 영상을 처리한다면, a, b가 1보다 큰 값을 가지는 경우, 영상 데이터는 확대가 되고, 1보다 작을 때에는 축소가 된 출력 데이터를 얻을 수 있다. 식 (4-1)에 의해 우리는 모든 화소 점(x,y)에 대해서 이 연산을 행하고, 출력 영상의 점(X,Y)의 농도 값에 입력 영상의 점(x,y)의 농도 값을 쓰면, 영상의 확대, 축소를 할 수 있다.

예제 프로그램은 (부록 1-a)와 같다. (부록 1-a)에서 영상배열 m_OpenImg와 m_ResultImg의 크기는 가로 256,세로 256의 화소이므로 첨자 x, y의 범위는 $-256/2<x<256/2$, $-256/2<y<256/2$ 가 된다. 그러나 x, y에 음수는 허락 되지 않기 때문에, 좌표(x, y)에 접근할 때는 (256/2,256)만 바이어스를 걸고, m_OpenImg[y+256/2][x+256/2]와 m_ResultImg[y+256/2][x+256/2]와 같이 접근할 수 있도록 한다. 또 계산의 결과가 영상 배열의 범위를 넘어가지 않도록 방지하고 있다. (부록 1-b)프로그램을 사용하여 2배로 확대했을 때와 1/2로 축소한 경우를 생각해 보자. 1/2축소는 좋지만 2배 확대한 그림은 조금 변한 영상이 된다.

이를 피하기 위해 "입력 영상을 기준으로 한 출력 영상의 mapping 방법(forward mapping method)"을 사용하지 않고, 출력 영상을 기준으로 해서 출력 영상의 화소가 입력 영상의 어떤 화소에 대응 관계(inverse mapping method)를 이용하는 것이 더 나은 결과가 나올 수 있다. 이는 inverse mapping 의 경우 각각의 출력영상의 화소에 그것의 변환된 입력영상의 화소를 가지고 채워 나가므로 채워지지 않는 화소는 생기지 않기 때문이다. 이 과정을 수행하기 위하여, 위 식의 역 변환을 생각해 보자. 그 역 변환은 다음과 같다.

$$x=X/a$$
$$y=Y/b \qquad (4-2)$$

출력 영상의 모든 화소(X,Y)에 대하여 역변환 계산하고, 대응하는 입력영상을 구해 이 화소의 농도 값을 쓰면, 위에서 언급한 현상은 일어나지 않을 것이다. 이와 같은 방법으로 확대, 축소를 하는 프로그램은 (부록 1-b) 같다.

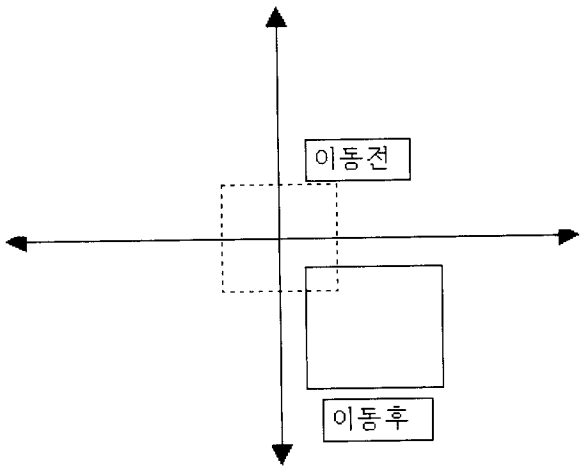
3.3 위치 변환

영상의 위치를 변환하는 방법은 아래 그림과 같이, 우측으로 x_0, 아래쪽으로 y_0가 움직이기 위한 변환식은 다음과 같다.

$$\begin{aligned} \mathbf{X} &= \mathbf{x} + \mathbf{x}_0 \\ \mathbf{Y} &= \mathbf{y} + \mathbf{y}_0 \end{aligned} \tag{4-3}$$

그 역 변환은 다음과 같다.

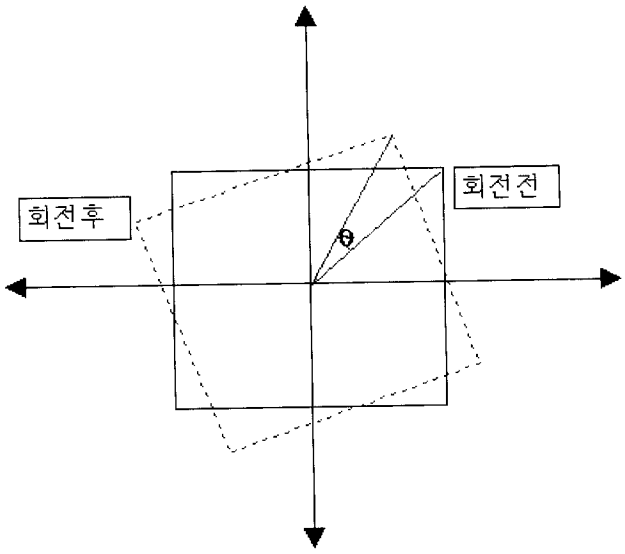
$$\begin{aligned} \mathbf{x} &= \mathbf{X} - \mathbf{x}_0 \\ \mathbf{y} &= \mathbf{Y} - \mathbf{y}_0 \end{aligned} \tag{4-4}$$



[그림 4.3] 위치변환

예제 프로그램은 (부록 1-c)와 같다.

4.4 회전 변환



[그림 4.4] 회전변환

[그림 4.4]는 정사각형의 모양을 θ만큼 회전했을 때의 모양이다. 회전되기 전의 좌표를 (x,y), 회전된 후의 좌표를 (X,Y)라고 했을 경우 영상을 반 시계 방향으로 θ도 회전한 식은 다음과 같다.

$$X = x \cdot \cos\theta + y \cdot \sin\theta$$
$$Y = -x \cdot \sin\theta + y \cdot \cos\theta \qquad (4\text{-}5)$$

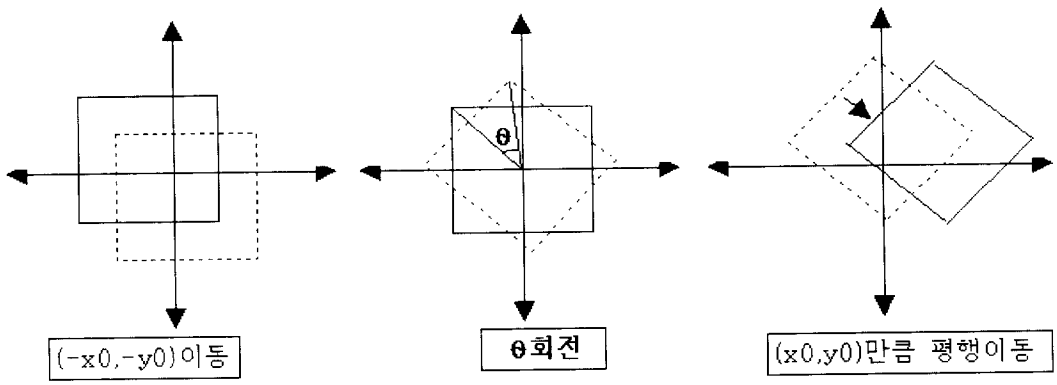
그 역 변환은 다음과 같다.

$$x = X \cdot \cos\theta - Y \cdot \sin\theta$$
$$y = X \cdot \sin\theta + Y \cdot \cos\theta \qquad (4\text{-}6)$$

예제 프로그램은 (부록 1-d)와 같다.

4.5 변환의 응용

위에서 설명한 확대, 축소, 이동, 회전을 조합하면, 여러 가지 형태가 가능하다. 지금까지의 설명은, 보통 원점을 중심으로 한 변형에 대한 설명이었지만 임의의 점을 중심으로 한 회전, 확대, 축소도 가능하다. 예를 들면, (x_0, y_0)을 중심으로 한 회전은, 영상을 먼저$(-x_0, -y_0)$이동하고, (x_0, y_0)을 원점으로 하여 θ도 회전시킨 후 (x_0, y_0)만큼 이동시키면 얻을 수 있다.



[그림 4.5] 회전 변환의 다른 응용

이 기법을 프로그램으로 살펴보면 앞에서 설명한 이동, 회전, 이동을 차례로 수행하는 과정이라고 할 수 있다. 이 기법은 처리 과정에서 주소(address)계산, 영상 데이터 접근, 농도 값 계산 등의 과정을 거치므로 많은 처리 시간이 필요하다. 주소(address)의 계산은 다음 식으로 얻을 수 있다.

$$\begin{aligned} X &= (x-x_0)\cos\theta + (y-y_0)\sin\theta + x_0 \\ Y &= -(x-x_0)\sin\theta + (y-y_0)\cos\theta + y_0 \end{aligned} \qquad (4\text{-}7)$$

주소 계산의 역 변환은 다음과 같다.

$$\begin{aligned} x &= (X-x_0)\cos\theta - (Y-y_0)\sin\theta + x_0 \\ y &= (X-x_0)\sin\theta + (Y-y_0)\cos\theta + y_0 \end{aligned} \qquad (4\text{-}8)$$

예제 프로그램은 (부록 1-e)와 같다.

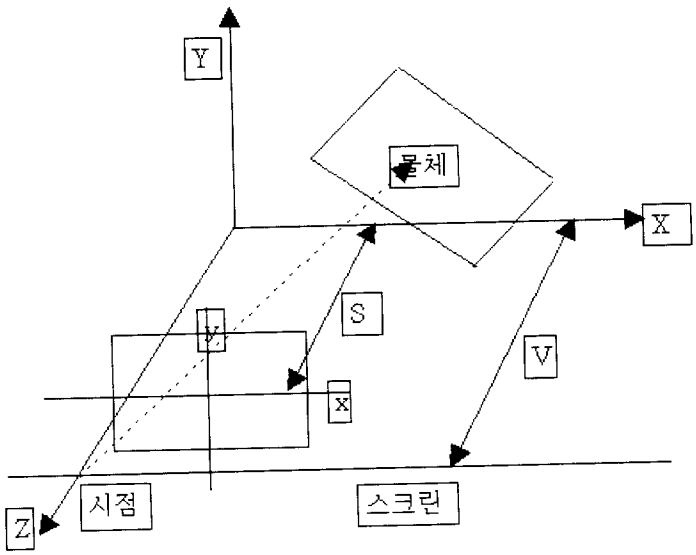
처리된 영상에 임의의 굴곡을 주지 않는 기하학적 변환을 affin 변환이라 한다. 위의 프로그램에서 보여주는 기하학적 변환을 2차원 affin 변환이라 한다. 2차원의 affin환 식은 다음과 같다.

$$X=ax+by+c$$
$$Y=dx+ey+f \qquad (4\text{-}9)$$

이 역 변환은 다음과 같다.

$$x=AX+BY+C$$
$$y=DX+EY+F \qquad (4\text{-}10)$$

위의 수식을 살펴보면, 파라미터는 다르지만 형태는 동일하다는 것을 알 수 있다. 앞에서 설명한 확대, 축소, 위치 이동, 회전의 식은 모두 위의 식에 포함되어 있다. 이 식들은 1차원의 다항식이지만 이것을 고차원의 다항식으로 변환시키면, 이장의 제일 앞에서 보여준 복잡한 기하학적 변환도 가능하다.(이러한 변환을 투시 변환이라 한다.) 이 기법을 이용하면,



[그림 4.6] 투시변환

[그림 4.6]은 그림을 그릴때, 원근을 묘사하는 것과 같은 효과를 얻을 수 있는데 물체를 시점으로부터 보고, 입체가 화면상에 촬영될 수 있는 투시 변환이다. 투시변환식은 다음과 같다.

$$X=(ax+by+c)/(px+qy+r)$$
$$Y=(dx+ey+f)/(px+qy+r) \qquad (4\text{-}11)$$

그 역 변환은 다음과 같다.

$$x=(AX+BY+C)/(PX+QY+R)$$
$$y=(DX+EY+F)/(PX+QY+R) \qquad (4\text{-}12)$$

여기에서, a,b,c와 A,B,C등은 변환 파라미터로서 눈의 위치, 화면의 위치, 크기에 의해서 변화한다. 이 파라미터들은 동차 좌표 표현을 이용한 행렬 연산을 행하여 간단히 구할 수 있다. 예제 프로그램은 (부록 1-f)와 같다.

4.6 동차 좌표표현

기하학적 변환은 행렬을 이용하여 훌륭하게 얻을 수 있다. 2차원 평면(X,Y)의 기하학적 변환은 2차원 벡터[X Y]와 2*2의 행렬로 표현할 수 있을 것같이 생각하기 쉽지만 이 방법으로는 이동을 표현할 수 없다. 그래서 이동 등도 동시에 취급하기 위해서는 가상의 차원 W을 추가하여, 3차원 벡터 [X Y W]와 3*3의 행렬을 보통 사용하고 있다. 이 3차원 공간의 좌표 (X,Y,W)을 (X,Y)의 동차 좌표라고 부르고 있다. 동차좌표의 개념을 이용하면 모든 Affin 변환을 행렬을 이용하여 쓸 수 있다. 이 동차 좌표의 표현에 의하면, Affin 변환은

$$[X \quad Y \quad W] = [x \quad y \quad 1]\begin{bmatrix} a & d & 0 \\ b & e & 0 \\ c & f & 1 \end{bmatrix} \tag{4-13}$$

확대, 축소는,

$$[X \quad Y \quad W] = [x \quad y \quad 1]\begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4-14}$$

이동은,

$$[X \quad Y \quad W] = [x \quad y \quad 1]\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ xo & y0 & 1 \end{bmatrix} \tag{4-15}$$

회전은,

$$[X \quad Y \quad W] = [x \quad y \quad 1]\begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4-16}$$

로서 구할 수 있고, 각각 본문중의 확대, 축소, 이동, 회전 변환의 식과 일치한다. 이 행렬들을 조합하면, 여러 가지 Affin 변환을 표현할 수 있다. 예를 들면, (x0,y0)을 중심으로 한 회전은,

$$[X \quad Y \quad W] = [x \quad y \quad 1] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -xo & -y0 & 1 \end{bmatrix} \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ xo & y0 & 1 \end{bmatrix} \quad (4\text{-}17)$$

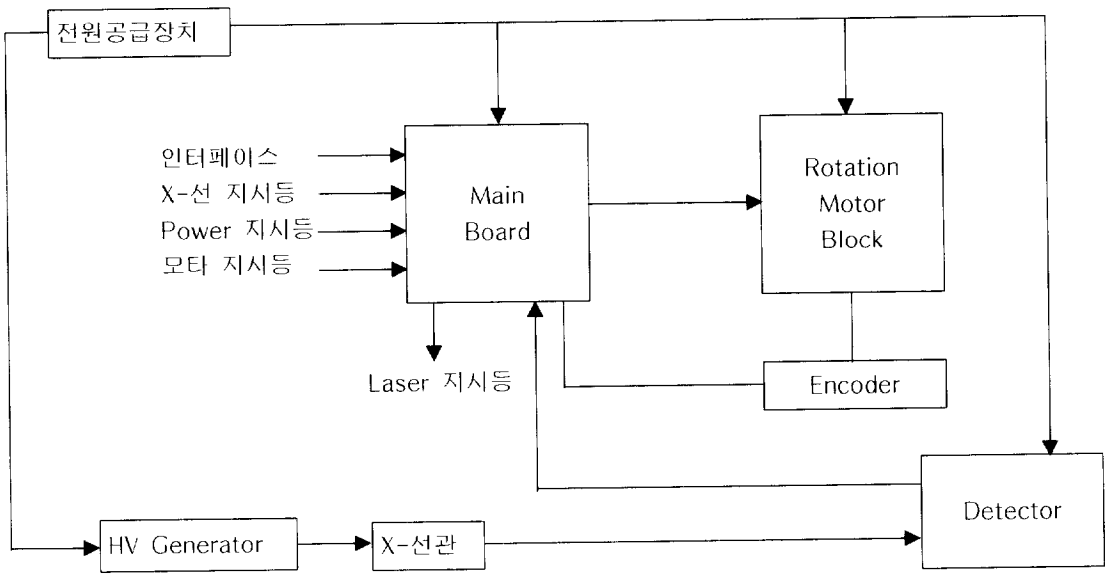
투시 변환 등의 3차원 공간의 변환의 경우는, 4차의 벡터와 4*4의 행렬을 이용하여 표현한다. 컴퓨터 그래픽 등에서 자주 이용되고 있다.

제 5 장 골밀도 진단기 설계 및 제작

5.1 동작원리

X선 고전압 발생기에서 60kvp까지 관전압을 높였을 때 X선관을 통하여 X선이 방출된다. X선관이 270도 회전하며 X선관을 통해 방출되는 X선이 환자의 발뒤꿈치 또는 팔목부위에 주사된다. X 선관의 1도에 1개의 상을 형성하며 X선관 및 검출기의 각도를 1도 변경하여 두 번째 주사를 하는 방식으로 각도를 변형시켜 270도를 회전하면서 주사한다. 한번 주사해서 얻은 검출기의 아날로그 신호를 받아 컴퓨터 프로그램(소프트웨어 BoneCT Ver1.0)을 통해 영상을 재구성한 후 모니터에 표시한다.

5.2 블록다이어그램

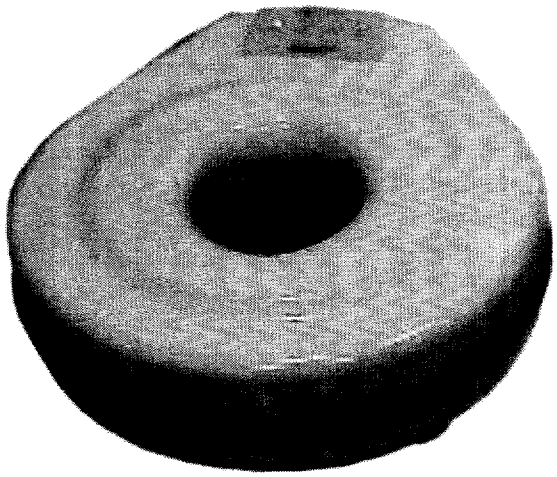


[그림 5.1] 골밀도 진단기의 블록도

5.3 골밀도 진단기의 설계

가) 외관사진

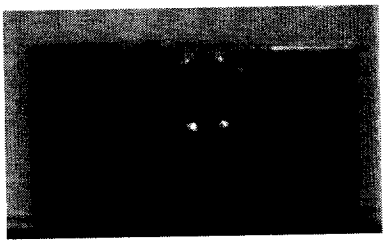
(1) 본체



[그림 5.2] 골밀도 진단기의 외관

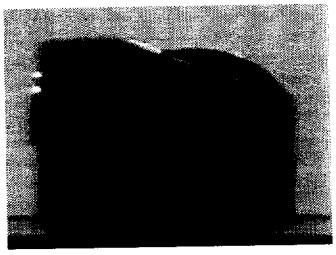
(2) X선관

-정면



(a)

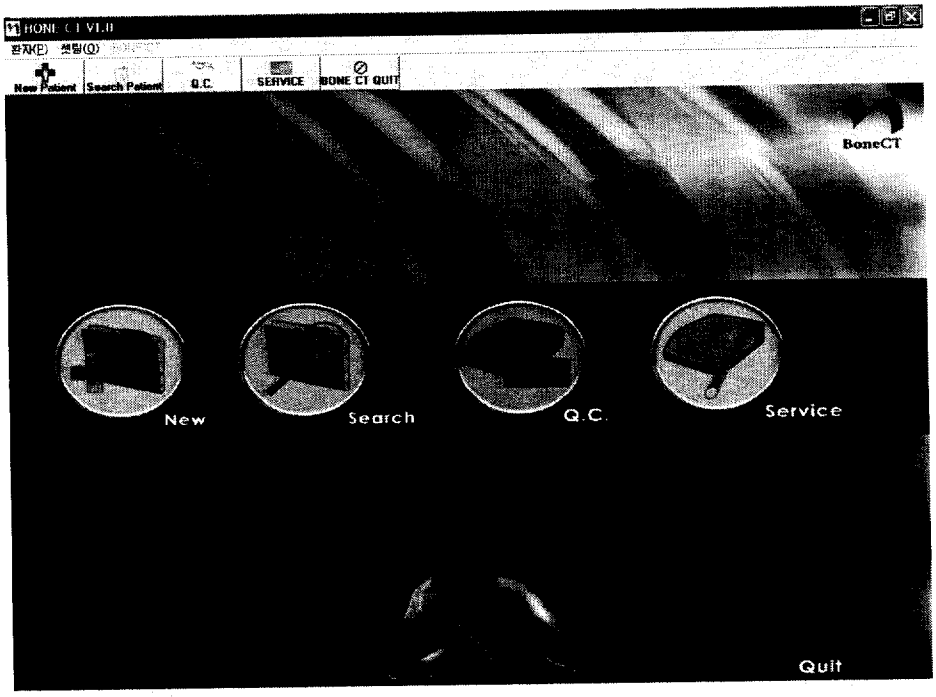
-측면



(b)

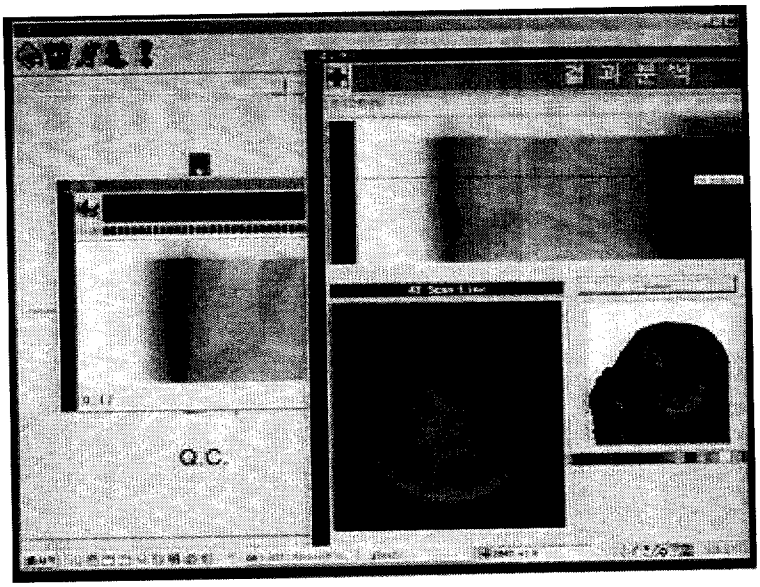
[그림 5.3] X-선관

5.4 메인화면



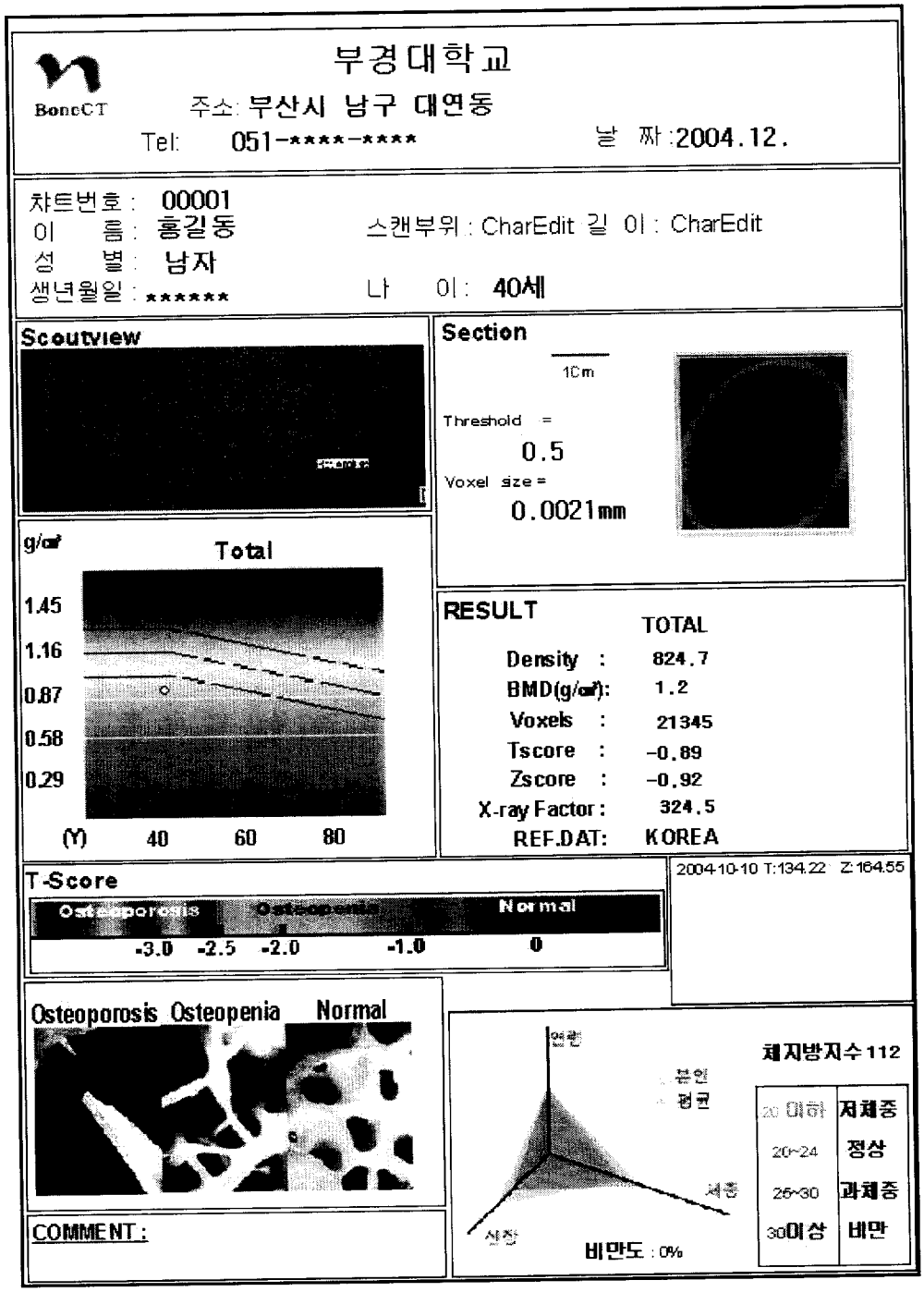
[그림 5.4] 메인화면

5.4 결과화면



[그림 5.5] 결과화면

5.4 최종 결과지


부경대학교
BoneCT
주소: 부산시 남구 대연동
Tel: 051-****-****
날 짜:2004.12.
차트번호 : 00001
이 름 : 홍길동
성 별 : 남자
생년월일 : ******
스캔부위 : CharEdit 길 이 : CharEdit
나 이 : 40세
Scoutview
Section
10m
Threshold =
0.5
Voxel size =
0.0021mm
g/㎠
Total
1.45
1.16
0.87
0.58
0.29
(Y) 40 60 80
RESULT
TOTAL
Density : 824.7
BMD(g/㎠): 1.2
Voxels : 21345
Tscore : -0.89
Zscore : -0.92
X-ray Factor : 324.5
REF.DAT: KOREA
T-Score
Osteoporosis
Osteopenia
Normal
-3.0 -2.5 -2.0 -1.0 0
2004-10-10 T:134.22 Z:164.55
Osteoporosis Osteopenia Normal
체지방지수 112
평균
서중
비만도 : 0%
20 이하 저체중
20~24 정상
25~30 과체중
30이상 비만
COMMENT :

제 6 장 결론

본 논문에서는 기존의 X-ray Film에 투영된 평면 이미지로는 해부학적인 위치나 상호관계를 진단하기 어렵다는 단점과 CT(Computerized Tomography)장비를 이용하여 골밀도값을 측정할때 에너지·시간·비용부담이 커지는 단점을 보완하여 Forearm과 Heel만으로 골밀도를 진단할 수 있는 기기를 설계하였다.

CT장비에서보다 골밀도 진단시에 환자에 대한 X-선 선량도 줄이면서, 단층촬영을 하였다. 본 논문의 설계·제작된 기기로 골밀도 데이타를 분석한 결과, 기존에 사용되고 있는 골밀도 진단기들의 골밀도 측정결과에포함된 피질골(Cotical Bone)과 Tissue성분을 완전히 제거하였으며, 골밀도 측정값의 정확도도 높일 수 있었다.

향후 아동들의 성장판을 입체적으로 분석하고, 성장 예측치와 기타 소아질병을 미연에 예측할 수 있는 분야, 피부표피 내측의 지방을 더욱 더 정밀히 계측하여 비만정도를 예측할 수 있는 분야 등 많은 분야에 전기·전자·컴퓨터 이론과 의학적 이론이 접목되어 고품질이 의료장비가 고안 되어, 인류와 국가 경쟁력도 향상되리라 본다.

참 고 문 헌

[1] Ross PD,Davis JW,Epstein RS,Wasnich RD. Pre-existing fractures and bone mass predict vertebral fracture incidence in women. Ann Intern Med. pp. 19~23,1991.

[2] Melton LJ III,Lane AW,Cooper C,Eastell R,O'Fallon WM,Riggs,BL,Prevalence and of Vertebral Deformities.Osteoporosis Int pp. 113~119, 1993.

[3] Davis JW, Grove JS, Wasnich RD, Ross PD. Spatial relationships between prevalent and incidence spine fractures. Bone. pp. 261,1999.

[4] Pouilles JM,Tremollieres F, Ribot C. Spine and femur densitometry after menopause : are both sides necessary in the assessment of risk of Osteoporosis . Calcif Tissue Int. pp. 344~347, 1993.

[5] 이문호, *C언어를 이용한 영상 신호 처리*, 大英社.pp. 116~142,1994.

[6] 정제창 역, *응용 MPEG*, 교보문고. pp. 26~27,1997.

[7] 이문호, *The Weighted Hadamard/Discrete Cosine Transform for Image Coding and Their Systolic Array Processing*, 도서출판 해외 마케팅. pp.11~13,1990.

[8] 이문호, *C언어 Matlab을 이용한 디지털필터설계*, 대영사. pp.34~37,1997.

[9] 이문호, *실용정보이론-오류정정이론의 기초와 응용*, 복두. pp.55~57,1998.

[10] 이문호, *實用 디지털 通信 (기초와 응용)*, 영일.pp. 122~127,1999.

부록 1 . 3장 예제 프로그램

1-a . 크기의 변환

```
        //forward mapping method
void CtransDoc::Expan(float SCALE_X, float SCALE_Y)
{
        int i,j,y_bnd,x_bnd;
        int xs = 256/2;
        int ys = 256/2;

        for(i=0;i<256;i++)          //initialize
        for(j=0;j<256;j++)
        m_ResultImg[i][j] = 0;

        for(i=-ys;i<ys;i++)
        {
                for(j=-xs;j<xs;j++)
                {
                        y_bnd = SCALE_Y*i;
                        x_bnd = SCALE_X*j;
                        //mapping the value on range from -256/2 to 256/2
                        if((y_bnd>=-ys)&&(y_bnd<ys)
                                &&(x_bnd>=-xs)&&(x_bnd<xs))
                        m_ResultImg[y_bnd+ys][x_bnd+xs] =
                        m_OpenImg[i+ys][j+xs];
                }
        }
}
```

1-b . 크기의 변환

```
//reverse mapping method
void CtransDoc::InverseMap(float SCALE_X, float SCALE_Y)
{
```

```
        int i,j,y_bnd,x_bnd;
        int xs = 256/2;
        int ys = 256/2;

        for(i=0;i<256;i++)          //initialize
                for(j=0;j<256;j++)
                        m_ResultImg[i][j] = 0;

//역함수를 취하고 반올림을 한다.
        for(i=-ys;i<ys;I++)
        {
                for(j=-xs; j<xs;j++)
                {
                        if(i>0) y_bnd =i/SCALE_Y + 0.5;
                        else y_bnd =i/SCALE_Y -0.5;
                        if(j>0) x_bnd =j/SCALE_X +0.5;
                        else x_bnd =j/SCALE_X -0.5;

                        //mapping the value on range from -256/2 to 256/2
                        if((y_bnd>=-ys)&&(y_bnd<ys)
                        &&(x_bnd>=-xs)&&(x_bnd<xs))
                        m_ResultImg[i+ys][j+xs] =
                        m_OpenImg[y_bnd+ys][x_bnd+xs];
                        else m_ResultImg[i+ys][j+xs] = 0;
                }
        }
}
```

1-c . 위치 변환

```
//입력영상을 이동하는 함수
void CtransDoc::Move(int X_axis, int Y_axis)
{
        int i,j,buf1,buf2;
        float x,y,p,q;
        int xs = 256/2;
```

```
        int ys = 256/2;
        int data;
        for(i=-ys;i<ys;i++)
        {
                for(j=-xs;j<xs;j++)
                {
                y =i-Y_axis;    //inverse
                x=j-X_axis;
                if(y>0)buf1 = y;          //positive
                else buf1 = y-1; //negative
                if(x>0)buf2 = x;                  //positive
                else buf2 = x-1;          //negative
                q = y-buf1;
                p = x-buf2;
                if((buf1>-ys)&&(buf1<ys)&&(buf2>=-xs)&&(buf2<xs))
                data = (1.0-q)*((1.0-p)*m_OpenImg[buf1+ys][buf2+xs]
                                +p*m_OpenImg[buf1+ys][buf2+1+xs])
                                +q*((1.0-p)*m_OpenImg[buf1+1+ys][buf2+xs]
                                +p*m_OpenImg[buf1+1+ys][buf2+1+xs]);
                else
                        data = 0;
                        //if the data is beyond to the range
                        if(data<0)        data =0;
                        //if the value of data is lower than zero
                        if(data>255)     data = 255;
                        //if the value of data is higher than 255
                        m_ResultImg[i+ys][j+xs] = data;
                        //store the value into image_out
                }
        }
}
```

1-d . 회전변환

```
//입력 영상을 회전시키는 함수
void CtransDoc::Rotation(float angle)
```

```
{
        int i,j,buf1,buf2;          //declaration
        float x,y,p,q;
        double r;
        float c,s;
        int xs = 256/2;
        int ys = 256/2;
        int data;

r = angle*3.141592/180.0;           //radian value
c = cos(r);                         //cosine value
s = sin(r);                         //sine value

for(i=-ys; i<ys;i++)
{
        for(j = -xs;j<xs;j++)
        {
                y=j*s+i*c;
                x=j*c-i*s;
                if(y>0)buf1=y; //positive
                else buf1 = y-1;        //negative
                if(x>0)buf2 = x;        //positive
                else buf2 = x-1;        //negative
                q=y-buf1;
                p=x-buf2;
                if((buf1>-ys)&&(buf1<ys)&&(buf2>=-xs)&&(buf2<xs))
                data=(1.0-q)*((1.0-p)*m_OpenImg[buf1+ys][buf2+xs]
                        +p*m_OpenImg[buf1+ys][buf2+1+xs])
                        +q*((1.0-p)*m_OpenImg[buf1+1+ys][buf2+xs]
                        +p*m_OpenImg[buf1+1+ys][buf2+1+xs]);
                else
                        data = 0;
                        if(data<0)      data = 0;
                        if(data>255)    data = 255;
                        m_ResultImg[i+ys][j+xs] = data;
                }}
```

```
}
```

1-d . Affin 변환

```
//영상의 복잡한 변환을 하는 함수(Affin 변환)
void CtransDoc::Affin(float deg,float zx, float zy, float px, float py)
{
        int i,j,buf1,buf2 //declaration
        float x,y,u,v,p,q;
        double r;
        float c,s;
        int xs = 256/2;
        int ys = 256/2;
        int data;
        r = deg*3.141592/180.0;//radian value
        c=cos(r)                        //cosine value
        s=sin(r)                        //sine value
        for(i=-ys; i<ys;i++)
        {
                for(j=-xs;j<xs;j++)
                {
                        v =i-py;
                //the value of v is to subtract the value of position
                        movement
                //from output position
                        u = j-px;
                //the value of u is to subtract the value of position
                        movement
                //from output position
                y=(u*s + v*c)/zy;       //divided by expansion ratio
                x=(u*c - v*s)/zx;

                if(y>0)buf1 = y;                //positive
                else buf1 = y-1;        //negative
                if(x>0)buf2 = x;                //positive
                else buf2 = x-1;        //negative
                q = y-buf1;
```

```
                p = x-buf2;

                if((buf1>-ys)&&(buf1<ys)&&(buf2>=-xs)&&(buf2<xs))

        data = (1.0-q)*((1.0-p)*m_OpenImg[buf1+ys][buf2+xs]
                +p*m_OpenImg[buf1+ys][buf2+1+xs])
                +q*((1.0-p)*m_OpenImg[buf1+1+ys][buf2+xs]
                +p*m_OpenImg[buf1+1+ys][buf2+1+xs]);
                else
                        data = 0;

                if(data<0)
                        data = 0;
                if(data>255)
                        data = 255;
                m_ResultImg[i+ys][j+xs] = data;
                        //store the data in image_out
                }
        }
}
```

1-f . 투시변환

```
//투시변환을 하는 함수
void CtransDoc::Trans3D(float zx,float zy,float p_x,float p_y,float p_z,float
rot_x,
                        float rot_y,float rot_z,float view,float screen)
{
        int I,j,buf1,buf2;              //declaration
        float x,y,w,p,q;
        float k[9];
        int xs = 256/2;
        int ys = 256/2;
        int data;
//compute the parameter value
float ma[4][4], mb[4][4], mc[4][4], k1, k2, k3, k4, k5, k6, k7, k8, k9;
```

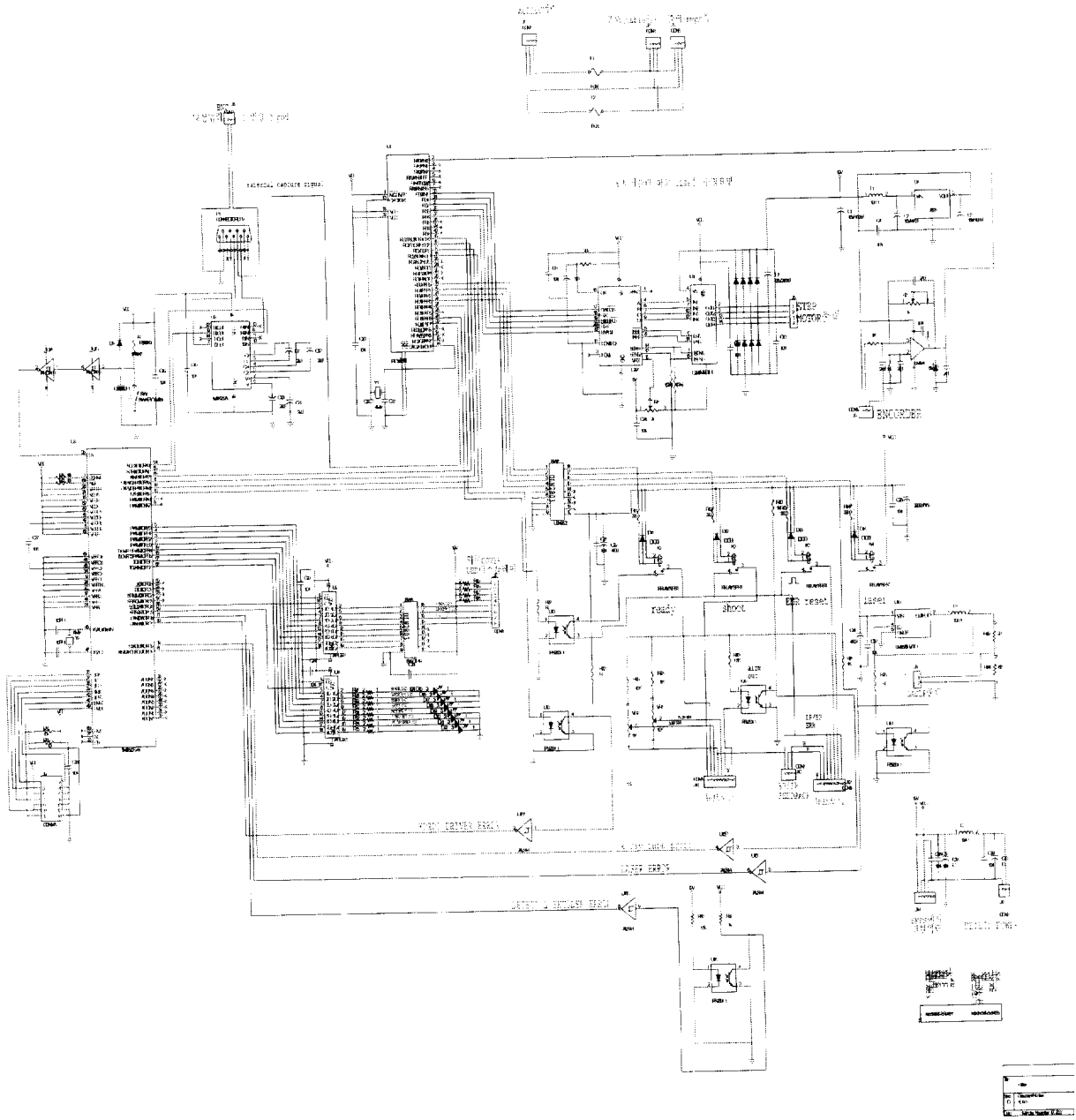
```
double u,v;
//rotation degree on x,y,z axis
u = rot_x*3.141592/180.0;
v = rot_y*3.141592/180.0;
w = rot_z*3.141592/180.0;

//namalization matrix
ma[0][0] = 1.0/xs; ma[0][1] = 0; ma[0][2] = 0; ma[0][3] = 0;
ma[1][0] = 0; ma[1][1] = -1.0/xs; ma[1][2] = 0; ma[1][3] = 0;
ma[2][0] = 0; ma[2][1] = 0; ma[2][2] = 1; ma[2][3] = 0;
ma[3][0] = 0; ma[3][1] = 0; ma[3][2] = 0; ma[3][3] = 1;

//matrix for expansion and contraction
mb[0][0] = zx; mb[0][1] = 0;mb[0][2] =0; mb[0][3] = 0;
mb[1][0] = 0; mb[1][1] = zy; mb[1][2] = 0; mb[1][3] = 0;
mb[2][0] = 0; mb[2][1] = 0; mb[2][2] = 1; mb[2][3] = 0;
mb[3][0] = mb[3][1] = 0; mb[3][2] = 0; mb[3][3] = 1;
Matrix(ma,mb,mc);

//matrix for shift
ma[0][0] = 1; ma[0][1] = 0; ma[0][2] = 0; ma[0][3] = 0;
ma[1][0] = 0; ma[1][1]  = 1; ma[1][2]  = 0; ma[1][3] = 0;
ma[2][0] = 0; ma[2][1] = 0; ma[2][2] = 1; ma[2][3] = 0;
ma[3][0] = p_x; ma[3][1] = p_y; ma[3][2] = p_z; ma[3][3] = 1;
Matrix(mc,ma,mb);

//matrix for rotation on z axis
mc[0][0] = cos(w); mc[0][1] = sin(w); mc[0][2] = 0; mc[0][3] = 0;
mc[1][0] = -sin(w); mc[1][1] = cos(w); mc[1][2] = 0; mc[1][3] = 0;
mc[2][0] = 0; mc[2][1] = 0; mc[2][2] = 1; mc[2][3] = 0;
mc[3][0] = 0; mc[3][1] = 0; mc[3][2] = 0; mc[3][3] = 1;
Matrix(mb,mc,ma);

//matrix for rotation on x axis
mb[0][0] = 1; mb[0][1] = 0; mb[0][2] = 0; mb[0][3] = 0;
mb[1][0] = 0; mb[1][1] = cos(u); mb[1][2] = sin(u); mb[1][3] = 0;
```

```
mb[2][0] = 0; mb[2][1] = -sin(u); mb[2][2] = cos(u); mb[2][3] = 0;
mb[3][0] = 0; mb[3][1] = 0; mb[3][2] = 0; mb[3][3] = 1;
Matrix(ma,mb,mc);

//matrix for rotation on y axis
ma[0][0] = cos(v); ma[0][1] = 0; ma[0][2] = sin(v); ma[0][3] = 0;
ma[1][0] = 0; ma[1][1] = 1; ma[1][2] = 0; ma[1][3] = 0;
ma[2][0] = -sin(v); ma[2][1] = 0; ma[2][2] = cos(v); ma[2][3] = 0;
ma[3][0] = 0; ma[3][1] = 0; ma[3][2] = 0; ma[3][3] = 1;
Matrix(mc,ma,mb);

//matrix to change view point
mc[0][0] = 1; mc[0][1] = 0; mc[0][2] = 0; mc[0][3] = 0;

mc[1][0] = 0; mc[1][1] = 1; mc[1][2] = 0; mc[1][3] = 0;
mc[2][0] = 0; mc[2][1] = 0; mc[2][2] = -1; mc[2][3] = 0;
mc[3][0] = 0; mc[3][1] = 0; mc[3][2] = view; mc[3][3] = 1;
Matrix(mb,mc,ma);

//matrix for perspect transform
mb[0][0] = 1; mb[0][1] = 0; mb[0][2] = 0; mb[0][3] = 0;
mb[1][0] = 0; mb[1][1] = 1; mb[1][2] = 0; mb[1][3] = 0;
mb[2][0] = 0; mb[2][1] = 0; mb[2][2] = 1/screen; mb[2][3] = 1/screen;
mb[3][0] = 0; mb[3][1] = 0; mb[3][2] = -1; mb[3][3] = 1;
Matrix(ma,mb,mc);

//normaization matrix
ma[0][0] = xs; ma[0][1] = 0; ma[0][2] = 0; ma[0][3] = 0;
ma[1][0] = 0; ma[1][1]  = -xs; ma[1][2] = 0; ma[1][3] = 0;
ma[2][0] = 0; ma[2][1] = 0; ma[2][2] = 1; ma[2][3] = 0;
ma[3][0] = 0; ma[3][1] = 0; ma[3][2] = 0; ma[3][3] = 1;
Matrix(mc,ma,mb);

//normaization matrix
ma[0][0] = xs; ma[0][1] = 0; ma[0][2] = 0; ma[0][3] = 0;
ma[1][0] = 0; ma[1][1] = -xs; ma[1][2] = 0; ma[1][3] = 0;
```

```
ma[2][0] = 0; ma[2][1] = 0; ma[2][2] = 1; ma[2][3] = 0;
ma[3][0] = 0; ma[3][1] =0; ma[3][2] = 0; ma[3][3] = 1;
Matrix(mc,ma,mb);

k1 = mb[0][3]; k2 = mb[1][3]; k3 = mb[3][3];
k4 = mb[0][0]; k5 = mb[1][0]; k6 = mb[3][0];
k7 = mb[0][1]; k8 = mb[1][1]; k9 = mb[3][1];

k[0] = k7*k2 - k8*k1; k[1] = k5*k1 - k4*k2; k[2] = k4*k8 - k7*k5;
k[3] = k8*k3 - k9*k2; k[6] = k9*k1 - k7*k3; k[4] = k6*k2 - k5*k3;
k[7] = k4*k3 - k6*k1; k[5] = k5*k9 - k8*k6; k[8] = k7*k6 - k4*k9;

for(i=-ys; i<ys;i++)
{
        for(j=-xs; j<xs;j++)
        {
                w = k[0]*j + k[1]*i + k[2];
                x = k[3]*j + k[4]*i + k[5];
                y = k[6]*j + k[7]*i + k[8];
                x = x/w;
                y = y/w;
                if(y>0) buf1 = y; else buf1 = y-1;
                if(x>0) buf2 = x; else buf2 = x-1;
                        q = y-buf1;
                        p = x-buf2;
                if((buf1>-ys)&&(buf1<ys)&&(buf2>-xs)&&(buf2<xs))
                        data = (1.0-q)*((1.0-p)*m_OpenImg[buf1 +
ys][buf2+xs]
                                + p*m_OpenImg[buf1 + ys][buf2 + 1 + xs])
                                + q*((1.0-p)*m_OpenImg[buf1+1+ys][buf2+xs]
                                +p*m_OpenImg[buf1+1+ys][buf2+1+xs]);
                else
                data = 0;

                        if(data<0) data =0;
                        if(data >255) data = 255;
```

```
                    m_ResultImg[i+ys][j+xs] = data;
                }
        }
}


//행렬의 곱을 계산하는 함수
void CtransDoc::Matrix(float ma[][4], float mb[][4], float mc[][4])
{
        int I,j,k;
        float p;
        for(i=0;i<4;i++)
        {
                for(j=0;j<4;j++)
                {
                        p=0;
                        for(k=0;k<4;k++) p=p+ma[i][k]*mb[k][j];
                        //multiplication
                        mc[i][j] = p;
                }
        }
}
```

2 . 회로도



3. 프로그램

```
#include <vcl.h>
#pragma hdrstop
#include <stdio.h>
#include "bdComm.h"
#include "Main.h"
#include "XCT.h"
#include "error.h"

TCommThread *Comm;// = new TCommThread(false);

#pragma package(smart_init)

__fastcallTCommThread::TCommThread(boolCreateSuspended):
TThread(CreateSuspended)
{
    Priority = tpLower;
    hCom = NULL;
    fCnt = FALSE;
}

__fastcall TCommThread::~TCommThread()
{
    CloseConnection();
}
void __fastcall TCommThread::Execute()
{
        //쓰래드에서는 데이터가 왔는지 감시하다가 들어온 데이터를 읽기만
          하면 된다

        DWORD dwEvtMask, dwLength, dwTotLen;
        OVERLAPPED os;
        dwTotLen = 0;
        memset(&os,0,sizeof(OVERLAPPED));
        os.hEvent=CreateEvent(NULL, TRUE, FALSE, NULL);
```

```
if(os.hEvent == NULL)
{
        Application->MessageBox
         ("Failed to create event for thread!","Com
          Error!",MB_ICONEXCLAMATION|MB_OK);
        return ;//(FALSE);
}

if(!SetCommMask(hCom,EV_RXCHAR))
{
        //AfxMessageBox("CommWatchProc Exit 3nd if");
        Application->MessageBox("CommWatchProcExit3nd
                if","ComError",MB_ICONEXCLAMATION|MB_OK );
        return ;//(FALSE);
}

 while(fCnt)
{
        dwEvtMask = 0;
         WaitCommEvent(hCom,&dwEvtMask,NULL);
                                        //데이터가 들어오때까지 대기

        if((dwEvtMask &EV_RXCHAR) == EV_RXCHAR)
        {
                                        //데이터가 들어 왔으면....
                do
                {
                    dwLength = SetReadData();
                        dwTotLen += dwLength;
                }       while(dwLength);

                if(dwTotLen > 0)
                {

          if (dwTotLen > 1000) goto ENDLABEL;
```

```
char buff[100];
//sprintf(buff,"Serial data : %d",*(REVDATA+0));
//BmdMenu->Label1->Caption = (AnsiString) buff;
ErrorCode = *(REVDATA+0);
if (ErrorCode > 9) ErrorCode = 0;

//BmdMenu->errorcheck = true;

for (int i=0;i<dwTotLen-5;i++)
{
    char buff[100];
    memcpy(buff,(REVDATA+i),9);
    if (strncmp(buff,"ch [ 1",6)==0)
    {
        memcpy(MainForm->TIMEVIEW,REVDATA,i);
        MainForm->TIMEVIEW[i] = 0x00;
    }
    if (strncmp(buff,"ch [16]  ",9)==0)
    {
        unsigned long temp=0;
        sscanf(REVDATA+i+9,"%x",&temp);
        MainForm->realChannel[15].Value = (double)temp;
    }
    if (strncmp(buff,"ch [ 1]  ",9)==0)
    {
        unsigned long temp=0;
        sscanf(REVDATA+i+9,"%x",&temp);
        MainForm->realChannel[0].Value = (double)temp;
    }
    if (strncmp(buff,"ch [ 2]  ",9)==0)
    {
        unsigned long temp=0;
        sscanf(REVDATA+i+9,"%x",&temp);
        MainForm->realChannel[1].Value = (double)temp;
    }
    if (strncmp(buff,"ch [ 3]  ",9)==0)
```

```
{
    unsigned long temp=0;
    sscanf(REVDATA+i+9,"%x",&temp);
    MainForm->realChannel[2].Value = (double)temp;
}
if (strncmp(buff,"ch [ 4]  ",9)==0)
{
    unsigned long temp=0;
    sscanf(REVDATA+i+9,"%x",&temp);
    MainForm->realChannel[3].Value = (double)temp;
}
if (strncmp(buff,"ch [ 5]  ",9)==0)
{
    unsigned long temp=0;
    sscanf(REVDATA+i+9,"%x",&temp);
    MainForm->realChannel[4].Value = (double)temp;
}
if (strncmp(buff,"ch [ 6]  ",9)==0)
{
    unsigned long temp=0;
    sscanf(REVDATA+i+9,"%x",&temp);
    MainForm->realChannel[5].Value = (double)temp;
}
if (strncmp(buff,"ch [ 7]  ",9)==0)
{
    unsigned long temp=0;
    sscanf(REVDATA+i+9,"%x",&temp);
    MainForm->realChannel[6].Value = (double)temp;
}
if (strncmp(buff,"ch [ 8]  ",9)==0)
{
    unsigned long temp=0;
    sscanf(REVDATA+i+9,"%x",&temp);
    MainForm->realChannel[7].Value = (double)temp;
}
if (strncmp(buff,"ch [ 9]  ",9)==0)
```

```
{
    unsigned long temp=0;
    sscanf(REVDATA+i+9,"%x",&temp);
    MainForm->realChannel[8].Value = (double)temp;
}
if (strncmp(buff,"ch [10]  ",9)==0)
{
    unsigned long temp=0;
    sscanf(REVDATA+i+9,"%x",&temp);
    MainForm->realChannel[9].Value = (double)temp;
}
if (strncmp(buff,"ch [11]  ",9)==0)
{
    unsigned long temp=0;
    sscanf(REVDATA+i+9,"%x",&temp);
    MainForm->realChannel[10].Value = (double)temp;
}
if (strncmp(buff,"ch [12]  ",9)==0)
{
    unsigned long temp=0;
    sscanf(REVDATA+i+9,"%x",&temp);
    MainForm->realChannel[11].Value = (double)temp;
}
if (strncmp(buff,"ch [13]  ",9)==0)
{
    unsigned long temp=0;
    sscanf(REVDATA+i+9,"%x",&temp);
    MainForm->realChannel[12].Value = (double)temp;
}
if (strncmp(buff,"ch [14]  ",9)==0)
{
    unsigned long temp=0;
    sscanf(REVDATA+i+9,"%x",&temp);
    MainForm->realChannel[13].Value = (double)temp;
}
if (strncmp(buff,"ch [15]  ",9)==0)
```

```
                {
                    unsigned long temp=0;
                    sscanf(REVDATA+i+9,"%x",&temp);
                    MainForm->realChannel[14].Value = (double)temp;
                }
                if (strncmp(buff,"RPM   ",5)==0)
                {
                    unsigned long temp=0;
                    sscanf(REVDATA+i+5,"%d",&temp);
                    MainForm->realChCount[0].Value = (double)temp;
                }
                if (strncmp(buff,"SPEED ",6)==0)
                {
                    unsigned long temp=0;
                    sscanf(REVDATA+i+6,"%d",&temp);
                    MainForm->realChCount[1].Value = (double)temp;
                }
            }
ENDLABEL:

MainForm->MonitorMemo->Lines->Strings[0].Pos("TIME");

                    }
                    dwTotLen = 0;
            }
        }

        CloseHandle(os.hEvent);
        return ;
}

bool TCommThread::OpenComm(String PortName, DWORD Baud, BYTE
Byte, BYTE Parity)
{                   //각 멤버 변수에 넘겨받은 변수들을 대입시킴으로서
포트설정을 끝낸다.
        DCB dcb;
```

```
COMMTIMEOUTS CommTimeOuts;
//---------------------------------------------------------------
osWrite.Offset = 0;
osWrite.OffsetHigh=0;
osRead.Offset=0;
osRead.OffsetHigh=0;
osRead.hEvent = CreateEvent(NULL,TRUE, FALSE,NULL);

if(osRead.hEvent == NULL)//이벤트가 발생하지 않으면
        return FALSE;

osWrite.hEvent = CreateEvent(NULL,TRUE, FALSE,NULL);

if(NULL == osWrite.hEvent)
{
        CloseHandle(osRead.hEvent);
        return FALSE;
}
```

```
if((hCom=CreateFile(PortName.c_str(),GENERIC_READ|GENERIC_WRITE,0,
NULL,
OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL|FILE_FLAG_OVERLAPPE
D,NULL))==(HANDLE)-1)
        {

                fCnt = FALSE;
                return(fCnt);
        }
        else
        {
        SetupComm(hCom, 4096, 4096);          //포트의 버퍼 크기 설정
        SetCommMask(hCom, EV_RXCHAR);//어떤 신호에 반응할것인가를
                                              //포트비우기
        PurgeComm(hCom,PURGE_TXABORT
```

```
                |PURGE_RXABORT|PURGE_TXCLEAR
                                        |PURGE_RXCLEAR);
            CommTimeOuts.ReadIntervalTimeout = 4;
            CommTimeOuts.ReadTotalTimeoutMultiplier=2;
            CommTimeOuts.ReadTotalTimeoutConstant=20;

            CommTimeOuts.WriteTotalTimeoutMultiplier=0;
            CommTimeOuts.WriteTotalTimeoutConstant=1000;
            SetCommTimeouts(hCom,&CommTimeOuts);
    }

    dcb.DCBlength = sizeof(DCB);
    GetCommState(hCom,&dcb);//dcb의 기본값은 받는다.
    dcb.BaudRate = Baud;
    dcb.ByteSize = Byte;
    dcb.Parity = Parity;
    dcb.StopBits = ONESTOPBIT;
    dcb.fOutxDsrFlow = 0;
    dcb.fDtrControl = DTR_CONTROL_ENABLE;
    dcb.fOutxCtsFlow = 0;
    dcb.fRtsControl = RTS_CONTROL_ENABLE;
    dcb.fInX = dcb.fOutX = 0;
    dcb.XonChar = ASCII_XON;
    dcb.XoffChar = ASCII_XOFF;
    dcb.XonLim = 100;
    dcb.XoffLim = 100;
    dcb.fBinary = TRUE;
    dcb.fParity = TRUE;
    dcb.fBinary = TRUE;
    dcb.fParity = TRUE;

if(SetCommState(hCom,&dcb))
{
     fCnt = TRUE;
     this->Resume();
}else
```

```
    {
        fCnt = FALSE;
        CloseHandle(hCom);
    }
    return(fCnt);
}


bool TCommThread::CloseConnection(void)     //컴포트 연결을 해제한다.
{
        fCnt = FALSE;
        SetCommMask(hCom,0);
        EscapeCommFunction(hCom,CLRDTR);
        PurgeComm(hCom,PURGE_TXABORT|PURGE_RXABORT
                                |PURGE_TXCLEAR|PURGE_RXCLEAR);

        // error control...
        if(hCom == NULL)
        {
                CloseHandle(hCom);
                return(FALSE);
        }
        else if(osRead.hEvent == NULL)
        {
                CloseHandle(osRead.hEvent);
                return(FALSE);
        }
        else if(osWrite.hEvent == NULL)
        {
                CloseHandle(osWrite.hEvent);
                return(FALSE);
        }
        return(TRUE);
}

DWORD    TCommThread::WriteCommBlock(LPCVOID    lpByte,    DWORD
```

```
dwToWrite)
{                       //보낼 데이터와 데이터의 크기를 인수로 받는다.
        DWORD dwWritten,dwError,dwErrorFlags;
        COMSTAT comstat;
        if(!WriteFile(hCom,lpByte,dwToWrite,&dwWritten,&osWrite))
        {
                if(GetLastError() == ERROR_IO_PENDING)
                {
                        //ERROR_IO_PENDING == 997
                        /* 읽을 문자가 남아 있거나 전송할 문자가
                           남아 있을 경우 Overapped IO의특성에
                           따라 ERROR_IO_PENDING에러메세지가
                           전달 된다.
                           timeout에 정해준 시간만큼 기다려준다.*/


 while(!GetOverlappedResult(hCom,&osWrite,&dwWritten,TRUE))
                        {
                                dwError = GetLastError();
                                if(dwError != ERROR_IO_INCOMPLETE)
                                {

                                    ClearCommError(hCom,&dwErrorFlags,
                                                          &comstat);
                                        break;
                                }
                        }
                }
                else
                {
                        dwWritten = 0;
                        ClearCommError(hCom, &dwErrorFlags, &comstat);
                }
        }
        return dwWritten;//실제로 쓰여진 바이트수를 리턴
}
```

```
DWORD TCommThread::SetReadData(void)   //읽은데이터를 버퍼에 저장
{
        DWORD       dwRead, dwError, dwErrorFlags;
        COMSTAT     comstat;
        ClearCommError(hCom, &dwErrorFlags, &comstat);
        //--- system queue에 도착한 byte수만 미리 읽는다.
        dwRead = comstat.cbInQue;
        //--> 시스템 큐에서 읽을 거리가 있으면..
        if(dwRead > 0)
        {
                //--> 버퍼에 일단 읽어들이는데.. 읽는데 실패했으면
                if(!ReadFile(hCom,REVDATA,
                                BUFFER_SIZE,&dwRead,&osRead) )
                {
                        //--> 읽을 거리가 남았있는지 확인
                        if (GetLastError() == ERROR_IO_PENDING)
                        {
                        //입출력중인 데이터가 있는 경우 timeouts에 정해준
                                                시간만큼 기다려준다.
                                while (! GetOverlappedResult(hCom,&osRead,
                                                &dwRead, TRUE))
                                {
                                        dwError = GetLastError();
                                if(dwErr!= ERROR_IO_INCOMPLETE)
                                        {

ClearCommError(hCom,&dwErrorFlags,&comstat);
                                                break;
                                        }
                                }
                        }
                        else
                        {
                                dwRead = 0;
```

```
                    ClearCommError(hCom,&dwErrorFlags,
                                                    &comstat);
                }
            }
        }
        //--> 실제 읽어들인 갯수를 리턴.
        return dwRead;
}
```