Renaud Sirdey

The Radon Transform

Éditions des Nik's News
www.niksnews.com/editions/

1998

Introduction

This document presents the Radon transform and a (non-exhaustive) set of methods dedicated to its inversion. We first focus on the problem of rotating a digital image. Secondly, we presents the Radon transform and its inverse followed by an introduction to some practical methods of inversion (filtered back projection, back projection and deconvolution,...). We then perform a (short) empirical study of the results given by these methods in terms of mean square error. Some proofs, code and figures are available in the appendix.

1 Rotating an image

1.1 Problems due to digital images

Rotating a continuous function of two variables by an angle θ is obviously done by using the following formula

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

where $\tilde{f}$ denotes the rotated version of f. The transformation can be expressed in matricial form as

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix} \tag{1}$$

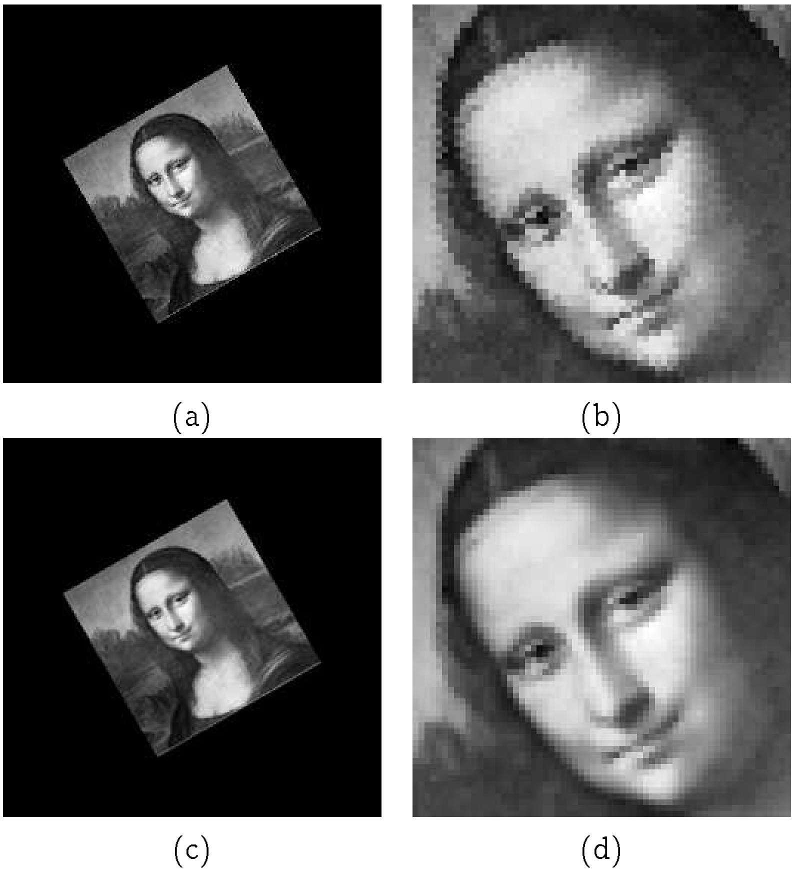
As soon as we are dealing with discrete functions (sampled images) some problems crop up. Let a g be a sampled version of f, we have[1]

$$\begin{aligned} g(x,y) &= f(x,y) \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} \delta(x-i, y-j) \\ &= \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} f(i,j)\delta(x-i, y-j) \end{aligned}$$

Therefore a digital image can be seen as set of spikes positionned at integer positions. Obviously, equation (1) does not define an integer-to-integer mapping except for special values of θ like $k\frac{\pi}{2}$, $k \in \mathbb{N}$. Hence the problem becomes: how to produce an sampled version of $\tilde{f}$ from a sampled version of f? In practice it is just possible to build some approximations.

[1] Assuming, without loss of generality, that $\Delta_x = \Delta_y = 1$.

Figure 1 Comparison between two rotation methods. (a) nearest neighbour, (b) zoom on (a), (c) linear interpolation, (d) zoom on (c).



1.2 The nearest neighbour method

The nearest neighbour method is the easiest method for rotating a digital image. For a pixel[2] $(i\ j)^T$ in the rotated image the algorithm consists in choosing the closest integer-indexed pixel to $(i'\ j')^T$ (computed by equation (1)) in the original image. Starting from the rotated image to the original one avoids the presence of gaps in the rotated image, since each pair $(i'\ j')^T$ has a nearest neighbour. For the reasons discussed in the previous section, this algorithm leads only to an approximation of the sampled version of $\tilde{f}$. Notice that the size of the resulting image is $2n \times 2n$ instead of $\sqrt{2}n \times \sqrt{2}n$, this avoid problems when applying FFT-based techniques for inverting the Radon transform (see page 7). See also figure 1.

[2]Note that i denotes the index along the columns and j the one along the rows.

1.3 Rotating using (bi)linear interpolation

An extension of the method discussed in the previous section consist in using (bi)linear interpolation for building the approximation of the sampled rotated function. This method has no more physical meaning than the previous one but visually the results are better as shown on figure 1.
A pixel with non-integer coordinates values (say $(i\ j)^T$) always fall in the square defined by

$$(\text{int}(i), \text{int}(j))^T,\ (\text{int}(i)+1, \text{int}(j))^T,\ (\text{int}(i), \text{int}(j)+1)^T,\ (\text{int}(i)+1, \text{int}(j)+1)^T$$

Let $\Delta_i = i - \text{int}(i)$ and $\Delta_j = j - \text{int}(j)$, we first compute the two values

$$v_1 = g(\text{int}(i), \text{int}(j)) + \Delta_i(g(\text{int}(i)+1, \text{int}(j)) - g(\text{int}(i), \text{int}(j)))$$

and

$$v_2 = g(\text{int}(i), \text{int}(j)+1) + \Delta_i(g(\text{int}(i)+1, \text{int}(j)+1) - g(\text{int}(i), \text{int}(j)+1))$$

The (bi)linear interpolated pixel value is then given by

$$v_1 + \Delta_j(v_2 - v_1)$$

2 Other methods

There exist a large number of methods for performing interpolation. Better visual results can be obtained by using (bi)cubic spline interpolation or (more tricky) wavelet interpolation. It is also interesting to note that the sinc interpolation as a physical meanning if the sampling theorem is satisfied, this assumption is likely to hold for real images because the instruments always have a cut-off spatial frequency.

3 The Radon transform

3.1 Definition

This section aims at defining the Radon transform. First of all, we consider a continuous function of two variables $f(x, y)$ and define a projection as being the following line integral

$$P(\theta, z) = \int_{L(z,\theta)} f(x, y) dl \tag{2}$$

The function $P(\theta, z)$ is the Radon transform of $f(x, y)$. According to the sampling property of δ, equation (2) can also be written as

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

The Radon transform defines a mapping from the (x, y) space to the (z, θ) space (also called the Radon space).

3.2 Reformulation

The two-dimensional δ-function (distribution) is defined as

$$\delta^2(\vec{u} - \vec{u}_0) = \delta(x - x_0)\delta(y - y_0),\ \vec{u} = (x\ y)^T$$

and can be expressed as follow (see page 10)

$$-\frac{1}{2\pi^2}\int_0^{\pi}\int_{-\infty}^{+\infty} \frac{1}{z - <\vec{\theta}, \vec{u}_0>} \frac{d}{dz}\delta(z - <\vec{\theta}, \vec{u}>)dzd\theta$$

3.3 The inverse Radon transform

Using the expression of δ^2 presented in the previous section (and derived page 10), the inverse Radon transform can be expressed as (see demonstration on page 12)

$$f(\vec{u}) = -\frac{1}{2\pi^2}\int_0^{\pi}\int_{-\infty}^{+\infty} \frac{1}{z - <\vec{\theta}, \vec{u}>} \frac{d}{dz}P(\theta, z)dzd\theta$$

4 The "practical" Radon transform

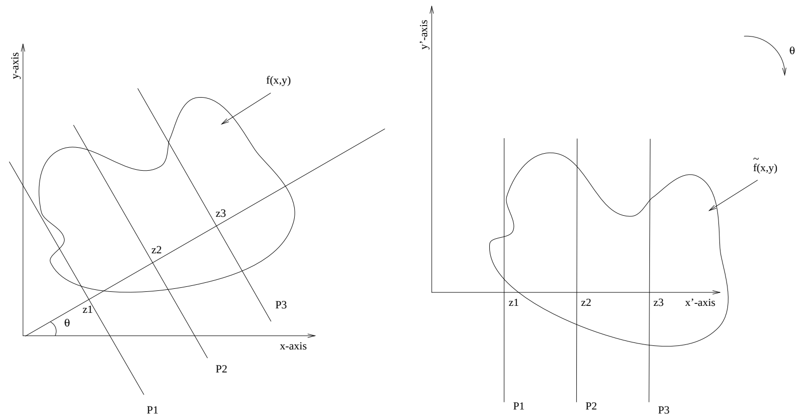
Let us consider the line integral which defines $P(\theta, z)$ (see equation (2)), if we consider $\tilde{f}$, the rotated version of f (continuous), by an angle $-\theta$ then equation (2) becomes a simple integral over the rows (y-axis) and can be expressed as

$$P(\theta, z) = \int_{-\infty}^{+\infty} \tilde{f}(z, y)dy$$

Hence, the Randon transform can be easily computed using a rotation procedure (see figure 2). Unfortunately, when we are dealing with digital images we cannot access a sampled version of the rotated continuous 2D-function from a sampled version of $f(x, y)$ (see page 3). Hence all the projections computed from a digital image are slightly corrupted. Obviously, because a computer has

a finite memory, the θ-axis must be discretized.
Therefore, from the indeterminacy theorem[3] and because of the discrete nature of the images we have to process, the Radon transform is not inversible. The next sections are dedicated to some techniques which allows to obtain an approximation of the 2D-object from a finite set of its projections. See also code page 12.

Figure 2 Calculation of the Radon tranform using a rotation procedure. (a) $P_i = \int_{L(\theta,z_i)} f(x,y)dl$, (b) $P_i = \int_{-\infty}^{+\infty} \tilde{f}(z_i, y')dy'$.



(a) (b)

5 Techniques of inversion

By studying the Fourier transform of the operation

$$\int_{-\infty}^{+\infty} \frac{1}{z - <\vec{\theta}, \vec{u}>} \frac{d}{dz} P(\theta, z) dz$$

it can be shown that the inverse Radon transform is given by

$$f(\vec{u}) = \frac{1}{2\pi} \hat{B}\hat{H} \frac{d}{dz} P(\theta, z) \tag{3}$$

where $\hat{B}$ denotes the back projection operator (described in the next section) and $\hat{H}$ denotes the Hilbert transform.

[3] *A function of compact support in two-dimensionnal Radon space is uniquely determined by an infinite set, but no finite set, of its projections.*

5.1 Back projection

The back projection method is involved in many techniques for inverting the Radon transform. Given a (infinite) set of projections the back projected image is defined as

$$b(x,y) = \int_0^{\pi} P(z,\theta)d\theta,\ z = x\cos\theta + y\sin\theta$$

Since, in practice, everything is discrete we only apply the following formula

$$f(i,j) = \sum_{k=0}^{n-1} P(i\cos k\Delta_\theta + j\sin k\Delta_\theta, k),\ \Delta_\theta = \frac{\pi}{n}$$

which leads to a smooth approximation of the original function. See figures 3, 4 (c) and 5 (c) (respectively pages 17, 18 and 19). See also code page 13.

5.2 Filtered back projection in the frequency domain

The filtered back projection in the frequency domain consists in appying the differentiation and the Hilbert transform (equation (3)) using Fourier-based techniques. The differentiation is performed using the well-known property of the Fourier transform

$$\frac{df}{dx} \Leftrightarrow i\xi\hat{f}(\xi) \tag{4}$$

and the Hilbert transform is performed (via the convolution theorem) by multiplying the Fourier transform of the projection by $-i\mathrm{sign}(\xi)$. The two operation are then done by multiplying $\hat{P}(\theta,\xi)$ by $|\xi|$ and the inverse Radon transform is computed by back projecting the processed projections. Now, recall that we are dealing with discrete signals, therefore the application of equation (4) is valid <u>only if the sampling theorem is satisfied</u>. See figures 4 (d) and 5 (d) (respectively pages 18 and 19). See also code page 14.

5.3 Filtered back projection in the "time" domain

Instead of using equation (4) for differentiating $P(\theta,z)$ according to z, we can use a finite difference scheme in the time domain. By using the Taylor expansion

$$f(x+h) = \sum_{k=0}^{\infty} \frac{h^k}{k!}\frac{d^kf}{dx^k}$$

one can obtain the following equation

$$\frac{df}{dx} \approx f(x+1) - f(x) \tag{5}$$

assuming $h = 1$ and neglecting the terms of order higher than 2. The Hilbert transform is then explicitly computed and the inverse Radon transform is also obtained by applying the back projection operator on the processed projections. Note that applying equation (5) gives an approximation of the derivative and therefore introduce some noise. See figures 4 (e) and 5 (e). See also code page 14.

5.4 Back projection and deconvolution

Instead of applying the "$\hat{B}\hat{H}\frac{d}{dz}$" scheme, one can derived a point spread function which links the object function to the back projection (without filtering). This gives the following equation

$$b(\vec{u}) = f(\vec{u}) \otimes h(\vec{u}), \ \ \hat{h}(\vec{\xi}) = \frac{1}{||\vec{\xi}||}$$

where b denotes the back-projected object. In the Fourier space we then have

$$\hat{f}(\vec{\xi}) = ||\vec{\xi}||\hat{b}(\vec{\xi})$$

which is (this time:-)) a well-conditionned filter. See figures 4 (f) and 5 (f) (respectively pages 18 and 19). See also code page 15.

6 Experimental results

The purpose of this section is to draw <u>quantitative and empirical</u> conclusions about the behaviour of the inversion methods discussed in the previous section. For the tests we have used the 128×128 image called mona.pgm. Here, we know the desired output of the inversion method since it is the image from which we have computed the Radon transform. Let f denotes the original image and $\tilde{f}$ the one given by a given inversion method. The following error measure

$$\Sigma = \frac{1}{n^2} \sum_{i,j} (f_{i,j} - \tilde{f}_{i,j})^2 \tag{6}$$

can be used for quantifying the loss of information due to

1. The discretization of the θ-axis.

2. The imperfections of the rotation method.

Figure 6 (page 20) shows the reconstruction given by the three inversion methods ($\Delta_\theta = \frac{\pi}{64}$) and table 1 gives the value of the expression given by equation (6). As shown on this figure (also compare figures 4 and 5, respectively pages 18 and 19) the reconstruction does not seem to be affected by the use of (bi)linear interpolation for computing the rotations. Figure 7 (page 21) presents the behaviour of Σ in function of Δ_θ, it also seems to reveal that the performances of a given method are not affected by the use of (bi)linear interpolation.
Fortunately, Σ has a strong tendency to decrease when Δ_θ decreases. However, it may localy increases (as Δ_θ decreases), this can (probably) be explained by the random nature of the information (noise) introduced by the rotation methods (some values of Δ_θ may introduce more noise than others). Figure 7 (d) (page 21) also reveals that the performances of the back projection and deconvolution method are less than those of the two other methods (which seems roughly equivalent).

Method	`filt.`	`hilb.`	`deconv.`
$\frac{1}{n^2}\sum_{i,j}(f_{i,j} - \tilde{f}_{i,j})^2$	596	452	892
Method	`filt. (i)`	`hilb. (i)`	`deconv. (i)`
$\frac{1}{n^2}\sum_{i,j}(f_{i,j} - \tilde{f}_{i,j})^2$	490	447	918

Table 1: Mean square error between the original image (f) and the images ($\tilde{f}$) of figure 6 (page 20).

A A convenient expression for δ^2

We start by the definition of the two dimensional δ function (distribution)

$$\delta^2(\vec{u} - \vec{u}_0) = \delta(x - x_0)\delta(y - y_0)$$

where $\vec{u} = (x\ y)^T$ and $\vec{u}_0 = (x_0\ y_0)^T$. Because $\delta(x - x_0) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} e^{i\xi(x-x_0)}d\xi$ we have

$$\begin{aligned}
\delta^2(\vec{u} - \vec{u}_0) &= \frac{1}{2\pi}\int_{-\infty}^{+\infty} e^{i\xi(x-x_0)}d\xi \frac{1}{2\pi}\int_{-\infty}^{+\infty} e^{i\lambda(y-y_0)}d\lambda \\
&= \frac{1}{4\pi^2}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} e^{i(\xi(x-x_0)-\lambda(y-y_0))}d\xi d\lambda \\
&= \frac{1}{4\pi^2}\int_{R^2} e^{i<\vec{\mu},\vec{u}-\vec{u_0}>}d\vec{\mu} \\
&= \frac{1}{4\pi^2}\int_{R^2} e^{i<\vec{\mu},\vec{u}>}e^{-i<\vec{\mu},\vec{u_0}>}d\vec{\mu} \qquad (7)
\end{aligned}$$

where $\vec{\mu} = (\xi, \lambda)^T$. Using the sampling property of the one-dimensional δ leads to

$$\int_{-\infty}^{+\infty} \delta(z - \frac{1}{|\vec{\mu}|} < \vec{\mu}, \vec{u}_0 >) e^{-i|\vec{\mu}|z} dz = e^{-i<\vec{\mu},\vec{u}_0>} \tag{8}$$

Hence by substituting equation (8) in equation (7) we can write

$$\delta^2(\vec{u} - \vec{u}_0) = \frac{1}{4\pi^2} \int_{R^2} e^{i<\vec{\mu},\vec{u}>} \int_{-\infty}^{+\infty} \delta(z - \frac{1}{|\vec{\mu}|} < \vec{\mu}, \vec{u}_0 >) e^{-i|\vec{\mu}|z} dz d\vec{\mu} \tag{9}$$

Now we use the fact that

$$\begin{aligned}
\int_{-\infty}^{+\infty} f(\xi, \lambda) d\xi d\lambda &= \int_0^{2\pi} \int_0^{+\infty} f(k\cos\theta, k\sin\theta) k dk d\theta \\
&= \int_0^{\pi} \int_{-\infty}^{+\infty} f(k\cos\theta, k\sin\theta) |k| dk d\theta \\
&= \int_0^{\pi} \int_{-\infty}^{+\infty} f(k\vec{\theta}) |k| dk d\theta
\end{aligned}$$

where $\vec{\theta} = (\cos\theta \;\; \sin\theta)^T = \frac{\vec{\mu}}{|\vec{\mu}|}$ and $k = |\vec{\mu}|$. Therefore equation (9) becomes

$$\begin{aligned}
\delta^2(\vec{u} - \vec{u}_0) &= \frac{1}{4\pi^2} \int_0^{\pi} \int_{-\infty}^{+\infty} e^{ik<\vec{\theta},\vec{u}>} \int_{-\infty}^{+\infty} \delta(z- < \vec{\theta}, \vec{u}_0 >) e^{-ikz} dz |k| dk d\theta \qquad (10) \\
&= \frac{1}{4\pi^2} \int_0^{\pi} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \delta(z- < \vec{\theta}, \vec{u}_0 >) e^{-ikz} e^{ik<\vec{\theta},\vec{u}>} k\,\mathrm{sign}(k) dz dk d\theta \\
&= \frac{1}{4\pi^2} \int_0^{\pi} \int_{-\infty}^{+\infty} \delta(z- < \vec{\theta}, \vec{u}_0 >) \int_{-\infty}^{+\infty} e^{-ik(z-<\vec{\theta},\vec{u}>)} k\,\mathrm{sign}(k) dk dz d\theta
\end{aligned}$$

We then study $\int_{-\infty}^{+\infty} e^{-ik(z-<\vec{\theta},\vec{u}>)} k\,\mathrm{sign}(k) dk$. From Fourier theory we have

$$\begin{aligned}
kf(k) &\Leftrightarrow i\frac{d\hat{f}}{dz} \\
\mathrm{sign}(k) &\Leftrightarrow \frac{2}{iz} \\
f(k)g(k) &\Leftrightarrow \frac{1}{2\pi} \int_{-\infty}^{+\infty} \hat{f}(t)\hat{g}(z-t) dt
\end{aligned}$$

Hence the previous integral becomes

$$2\int_{-\infty}^{+\infty} \frac{1}{z-t} \frac{d}{dt} \delta(t- < \vec{\theta}, \vec{u} >) dt \tag{11}$$

By substituting equation (11) into equation (10) we have

$$\delta^2(\vec{u} - \vec{u}_0) = \frac{2}{4\pi^2} \int_0^{\pi} \int_{-\infty}^{+\infty} \delta(z- < \vec{\theta}, \vec{u}_0 >) \int_{-\infty}^{+\infty} \frac{1}{z-t} \frac{d}{dt} \delta(t- < \vec{\theta}, \vec{u} >) dt dz d\theta$$

$$= \frac{1}{2\pi^2}\int_0^{\pi}\int_{-\infty}^{+\infty}\frac{d}{dt}\delta(t-<\vec{\theta},\vec{u}>)\int_{-\infty}^{+\infty}\delta(z-<\vec{\theta},\vec{u}_0>)\frac{1}{z-t}dzdtd\theta$$
$$= -\frac{1}{2\pi^2}\int_0^{\pi}\int_{-\infty}^{+\infty}\frac{1}{t-<\vec{\theta},\vec{u}_0>}\frac{d}{dt}\delta(t-<\vec{\theta},\vec{u}>)dtd\theta \qquad (12)$$

B Derivation of the inverse Radon transform

According to the sampling property of δ^2 we have

$$f(\vec{u}_0) = \int_{R^2} f(\vec{u})\delta^2(\vec{u}-\vec{u}_0)d\vec{u}$$

Using the form of δ^2 derived in the previous section leads to

$$\begin{aligned} f(\vec{u}_0) &= -\frac{1}{2\pi^2}\int_{R^2} f(\vec{u})\int_0^{\pi}\int_{-\infty}^{+\infty}\frac{1}{t-<\vec{\theta},\vec{u}_0>}\frac{d}{dt}\delta(t-<\vec{\theta},\vec{u}>)dtd\theta d\vec{u} \\ &= -\frac{1}{2\pi^2}\int_0^{\pi}\int_{-\infty}^{+\infty}\frac{1}{t-<\vec{\theta},\vec{u}_0>}\frac{d}{dt}\int_{R^2} f(\vec{u})\delta(t-<\vec{\theta},\vec{u}>)d\vec{u}dtd\theta \quad (13)\end{aligned}$$

As soon as we define $P(\theta,t)$ as $\int_{R^2} f(\vec{u})\delta(t-<\vec{\theta},\vec{u}>)d\vec{u}$ (see equation (2) on page 5), equation 13 becomes

$$f(\vec{u}_0) = -\frac{1}{2\pi^2}\int_0^{\pi}\int_{-\infty}^{+\infty}\frac{1}{t-<\vec{\theta},\vec{u}_0>}\frac{d}{dt}P(\theta,t)dtd\theta$$

and denotes the inverse Radon transform.

C Code

C.1 Radon transform

```
float **radon_2d(const float **in,int n,int ntheta,int *m,
    float**(*rot)(const float**,int,float,int*))
{
    float **out,**buf;
    register float theta_step,theta=.0;
    register int k,i,j;
    theta_step=M_PI/((float)ntheta);
    buf=(*rot)((const float**)in,n,.0,m);
    out=falloc_2d(*m,ntheta); /* rows => angle, columns => distance */
    for(k=0;k<ntheta;k++)
```

```
    {
#ifdef _UNIRAS
        uimage(buf,*m,*m,3,1,0);
        printf("\nroto scanner in action: #-- %f ",theta);
        fflush(stdout);
#endif
/* sum over the rows */
        for(j=0;j<*m;j++)
            for(i=0;i<*m;i++)
                out[k][i]+=buf[i][j];
        ffree_2d(buf,*m);
        theta+=theta_step;
        buf=(*rot)((const float**)in,n,-theta,m);
    }
    ffree_2d(buf,*m);
    return(out);
}
```

C.2 Back projection

```
float **backproj_2d(const float **in,int ntheta,int m)
{
    float **out;
    register float theta_step;
    register int k,i,j;
    theta_step=M_PI/((float)ntheta);
    out=falloc_2d(m,m);
    for(i=0;i<m;i++)
    {
        printf("\nroto projector in action: #-- %d ",i);
        fflush(stdout);
        for(j=0;j<m;j++)
            for(k=0;k<ntheta;k++)
            {
                register float theta,z;
                theta=((float)k)*theta_step;
z=((float)(i-(m>>1)))*cos(theta)+((float)(j-(m>>1)))*sin(theta)+(float)(m>>1);
                out[i][j]+=(ROUND(z)>=0&&ROUND(z)<m?in[k][ROUND(z)]:.0);
            }
```

```
    }
    return(out);
}
```

C.3 Filtered back projection (frequency domain)

```
float **filtbackproj_2d(float **in,int ntheta,int m)
{
    float **out,*img;
    unsigned int i,j;
    assert((img=(float*)malloc(m*sizeof(float)),img));
    for(i=0;i<ntheta;i++)
    {
        memset(img,0x0,m*sizeof(float));
        assert(fft1d(in[i],img,m,-1)==1);
        for(j=0;j<(m>>1);j++)
        {
            register float k=((float)((m>>1)-j))/((float)(m>>1));
            in[i][j]*=k;
            in[i][m-j-1]*=k;
            img[j]*=k;
            img[m-j-1]*=k;
        }
        assert(fft1d(in[i],img,m,1)==1);
    }
    free(img);
    return(backproj_2d((const float**)in,ntheta,m));
}
```

C.4 Filtered back projection ("time" domain)

```
float **hilbbackproj_2d(float **in,int ntheta,int m)
{
    float **out,*img;
    unsigned int i,j;
    assert((img=(float*)malloc(m*sizeof(float)),img));
    for(i=0;i<ntheta;i++)
    {
/* FIR differentiation */
```

```
        for(j=m-1;j>0;j--)
            in[i][j]-=in[i][j-1];
/* Hilbert transform */
        memset(img,0x0,m*sizeof(float));
        assert(fft1d(in[i],img,m,-1)==1);
        memset(in[i],0x0,(m>>1)*sizeof(float));
        memset(img,0x0,(m>>1)*sizeof(float));
        assert(fft1d(in[i],img,m,1)==1);
        memcpy(in[i],img,m*sizeof(float));
    }
    free(img);
    return(backproj_2d((const float**)in,ntheta,m));
}
```

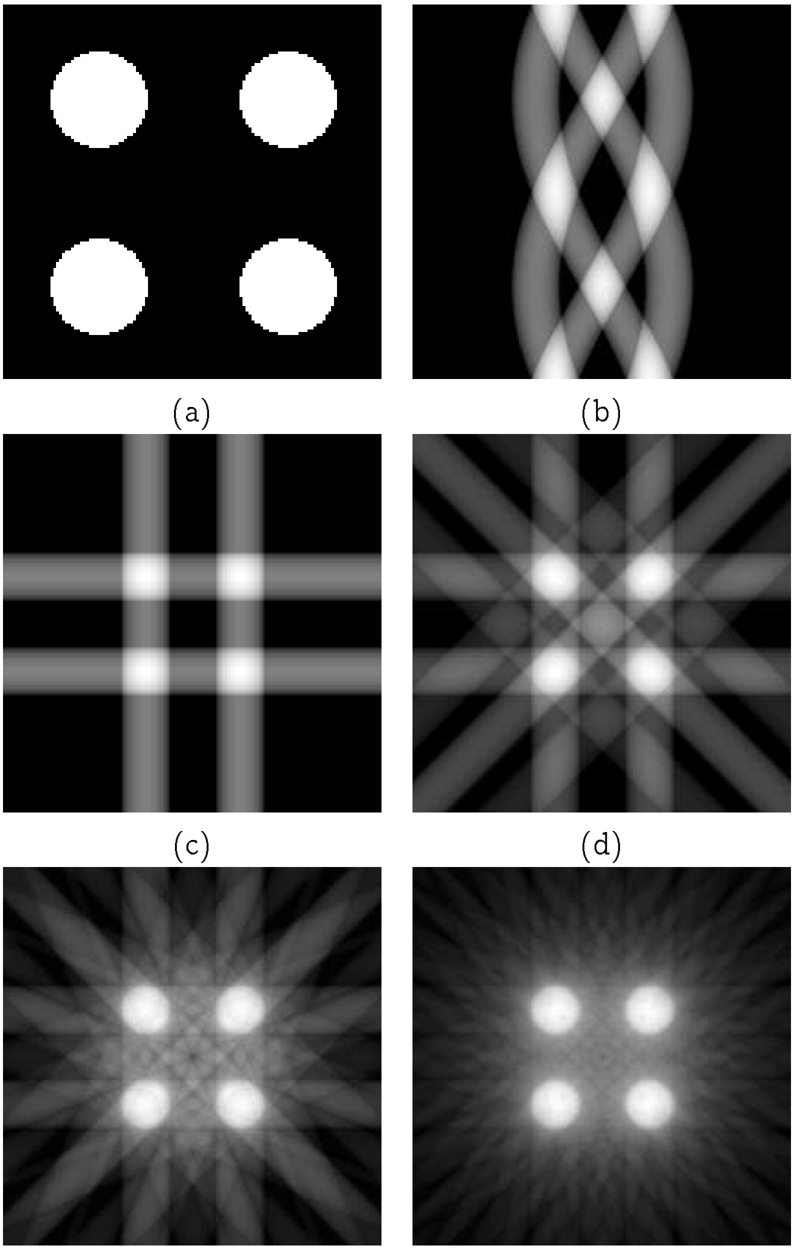
C.5 Back projection and deconvolution

```
float **decbackproj_2d(const float **in,int ntheta,int m)
{
    float **out,**img;
    unsigned int i,j;
    out=backproj_2d(in,ntheta,m);
    img=falloc_2d(m,m);
    assert(fft2d(out,img,m,-1)==1);
    for(i=0;i<m>>1;i++)
        for(j=0;j<m>>1;j++)
        {
            register float k=(float)sqrt((double)(SQR((m>>1)-i)+SQR((m>>1)-j)));
            out[i][j]*=k; img[i][j]*=k;
            out[i][m-j-1]*=k; img[i][m-j-1]*=k;
            out[m-i-1][j]*=k; img[m-i-1][j]*=k;
            out[m-i-1][m-j-1]*=k; img[m-i-1][m-j-1]*=k;

        }
    assert(fft2d(out,img,m,1)==1);
    ffree_2d(img,m);
    return(out);
}
```

D Figures

Figure 3 (a) Original image: `circles.pgm` (128×128), (b) Radon transform of (a) (rotations performed by the nearest neighbour technique). Reconstruction of (a) by back projection (images are 256×256) (c) $\Delta_\theta = \frac{\pi}{2}$, (d) $\Delta_\theta = \frac{\pi}{4}$, (e) $\Delta_\theta = \frac{\pi}{8}$, (f) $\Delta_\theta = \frac{\pi}{16}$.

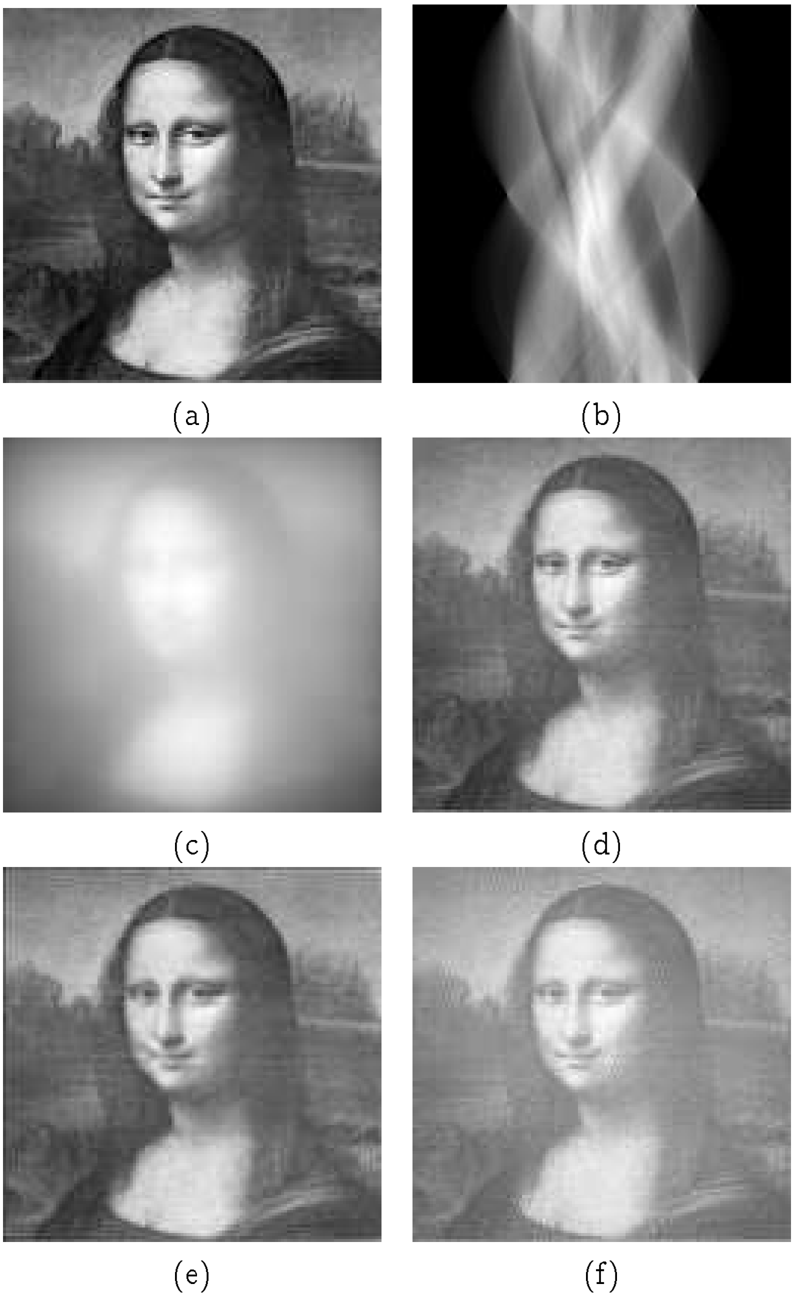


(a) (b)

(c) (d)

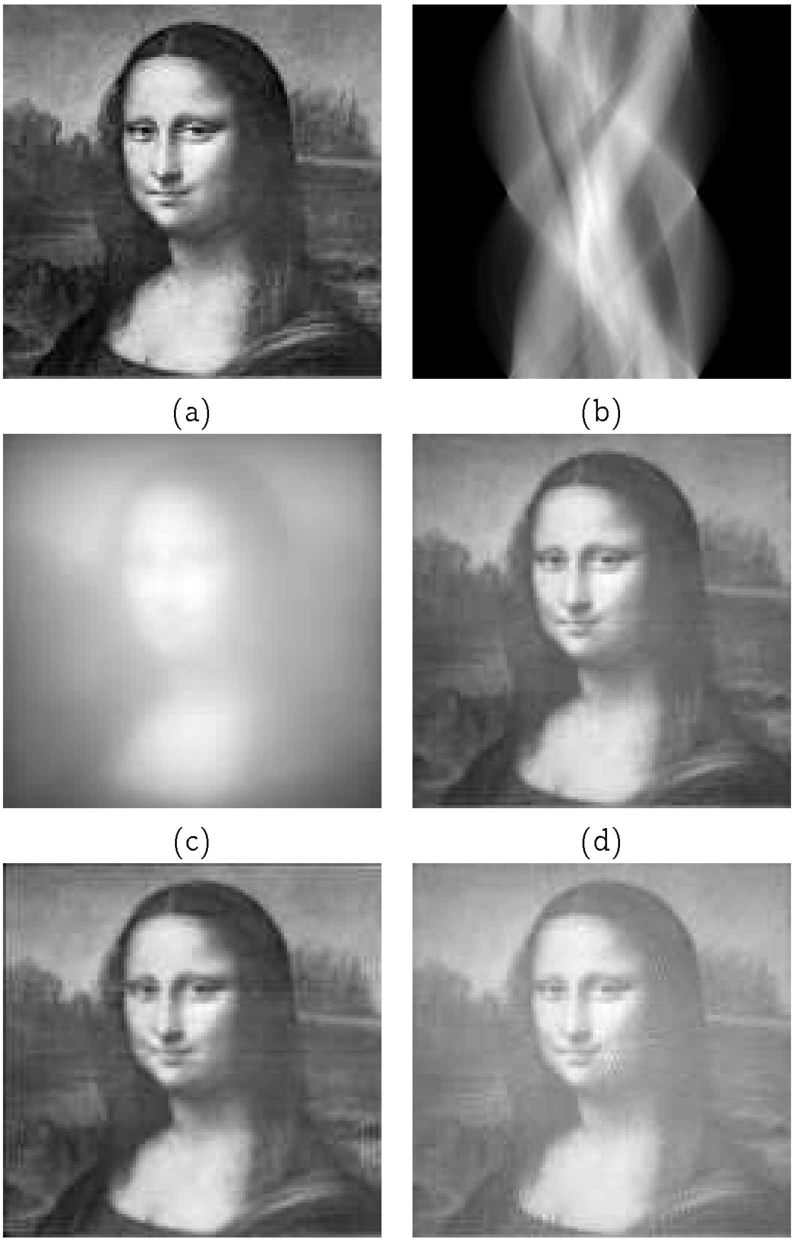
(e) (f)

Figure 4 (a) Original image: `mona.pgm`, (b) Radon transform of (a) ($\Delta_\theta = \frac{\pi}{256}$, rotations performed by the nearest neighbour technique). Reconstruction of (a) using (c) back projection, (d) filtered back projection in the frequency domain, (e) filtered back projection in the "time" domain, (f) back projection and deconvolution.



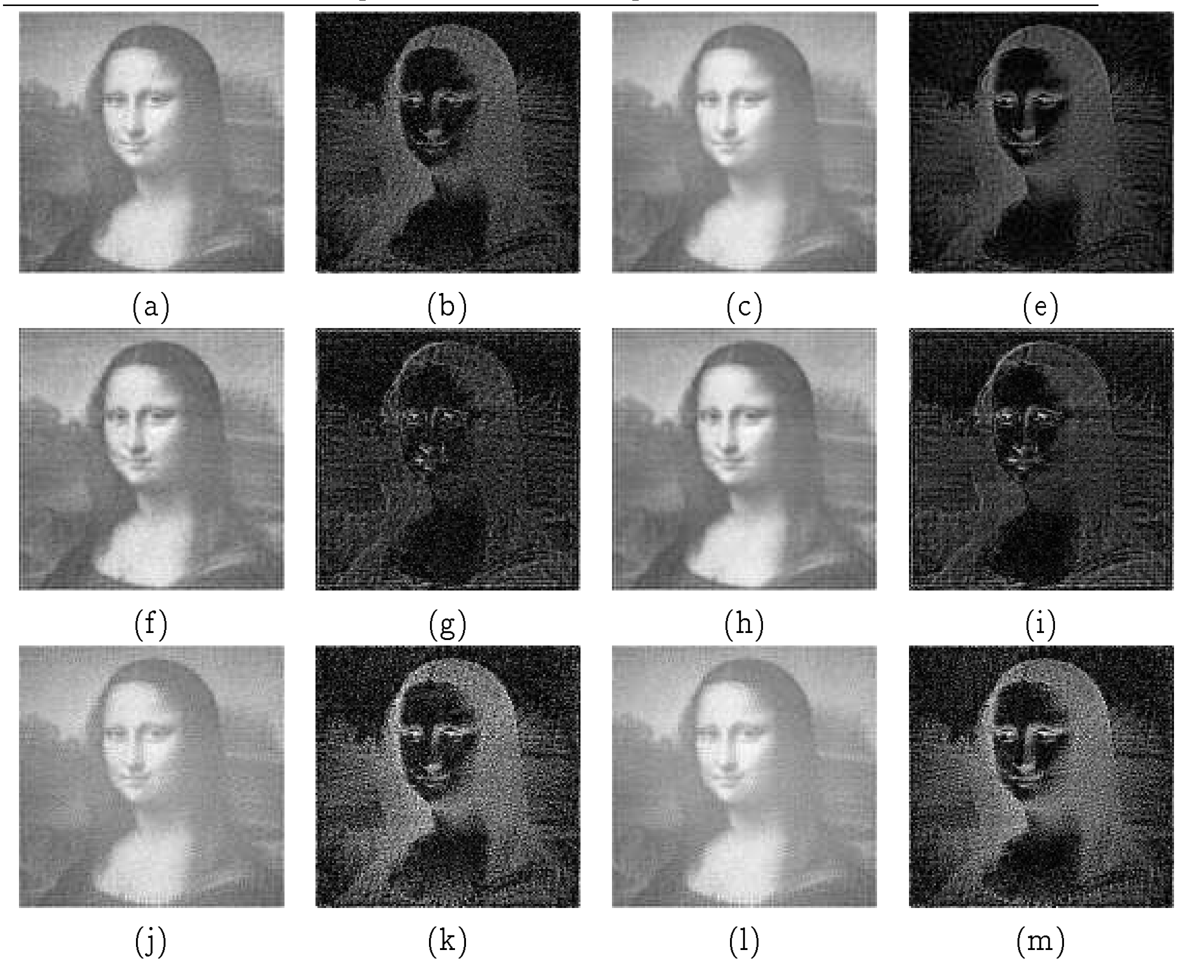
(a) (b) (c) (d) (e) (f)

Figure 5 Same as figure 4 but the rotations have been computed via the (bi)linear interpolation scheme.



(a) (b) (c) (d) (e) (f)

Figure 6 (a), (f) and (j) reconstruction of `mona.pgm` by (respectively) filtered back projection in the frequency domain, filtered back projection in the "time" domain, back projection and deconvolution ($\Delta_\theta = \frac{\pi}{64}$, no interpolation). (b), (g) and (k) absolute value of the error between (respectively) (a) and the original image, (f) and the original image, (j) and the original image. (c), (h), (l) (respectively) same as (a), (f) and (j) but using the interpolation scheme for computing the rotation. (e), (i) and (m) same as (repsectively) (b), (g) and (k) but for (c), (h) and (l). Note that the grey levels of (b), (g), (k), (e), (i) and (m) have been independantly rescaled to the interval $[0, 255]$, therefore comparing the intensity of these images has no meanning. See also text and table 1.

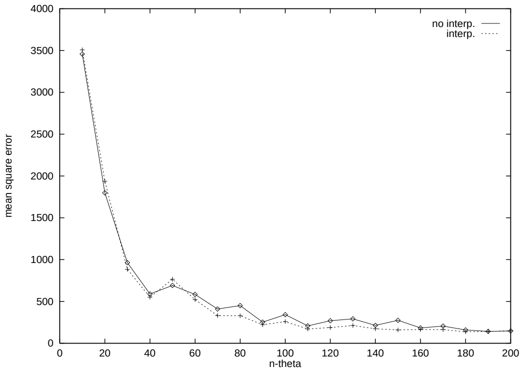


(a) (b) (c) (e)

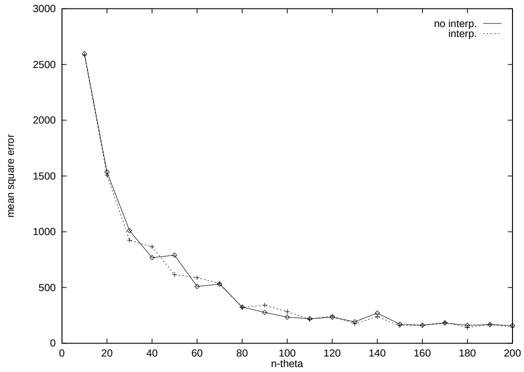
(f) (g) (h) (i)

(j) (k) (l) (m)

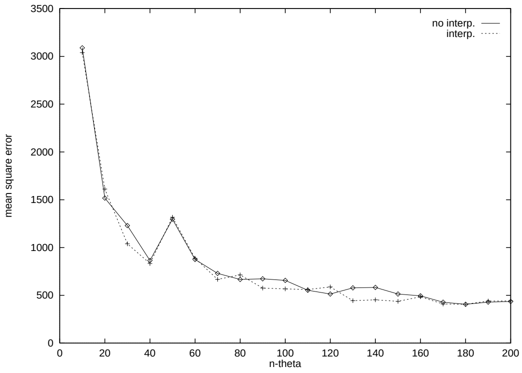
Figure 7 Mean square error $(\frac{1}{n^2}\sum_{i,j}(f_{i,j}-\tilde{f}_{i,j})^2)$ in function of $\frac{\pi}{\Delta_\theta}$. (a) filtered back projection in the frequency domain, (b) filtered back projection in the "time" domain, (c) back projection and deconvolution. (d) comparison of the three methods (no interpolation).



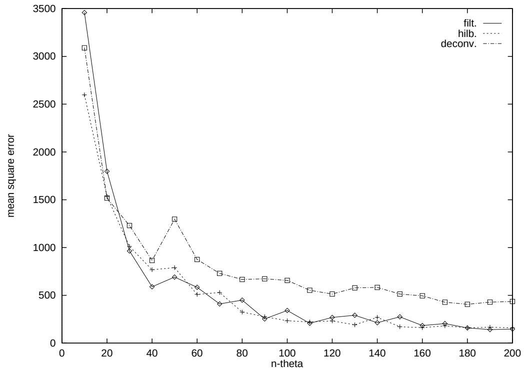
(a)



(b)



(c)



(d)