OPTIMIZED ALGORITHMS IN NUMERICAL CALCULATION OF THE RADON TRANSFORM

LEVON STEPANYAN

Department of Industrial Engineering and System Management, American University of Armenia, Yerevan, Armenia
E-mail: levon_stepanyan@edu.aua.am

Abstract - A common task in digital image processing is image reconstruction. One of the most common methods used in image reconstruction is the inverse Radon transform. The Radon Transform is an integral transform, which consists of the integral of a function over straight lines in different angles. One of its most common uses is the CT, when they use some beams to scan the object from different directions, and then reconstruct the inner and outer structure of the object. In this paper we are going to apply discrete Radon and inverse Radon transform for grayscale images of dimensions 512 x 512. Later generalize the transform for 3-channel RGB images, by doing the radon transform channel by channel. In this paper we propose an algorithm to implement the Radon and Inverse Radon transform by using parallel beam (back)projections.

Keywords - Radon Transform, discrete Radon, image reconstruction, sinogram,laminogram, Inverse Radon Transform, Ramp Filter.

I. INTRODUCTION

The two-dimensional Radon Transform was introduced in 1917 by Johann Karl August Radon, along with a formula for the inverse transformation. This transform was also given for the three-dimensional case (when the integral is taken over planes), by Radon himself. And was later generalized to higher-dimensional Euclidean spaces.

The Radon Transform has been very helpful in image processing, mainly image Reconstruction. Most of the classical image reconstruction techniques are using Inverse Radon Transform or a generalized version of Radon Transform. Image reconstruction by these methods allows to determine the inner structure without physically breaching the exterior. This is one of the main reasons that the Radon Transform has a wide variety of applications, such as tomography, ultrasound, x-ray,nuclear magnetic resonance imaging, optics, stress analysis, geophysics and many others.

The reconstruction problem is straightforward and can be explained in an intuitive manner. Consider the following image.

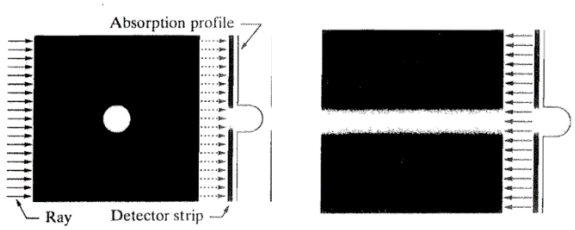


Fig.1. Radon Transform from one angle of a uniform circle.

Suppose this image is a cross section of a cat. The black region is soft tissue with small absorption properties while white circle is a tumor with higher absorption rates. Now suppose there are n equally spaced sources of parallel x-ray beams and also n detectors on the other side as shown in the Figure 1. We are shooting several parallel x-ray beams, the detectors on the receive an attenuated signal called absorption profile. Each point of which be treated as a sum of absorption values across one - corresponding line. We'll refer to this kind of sum as raysum. At this point we only have a 1D absorption signal. So, we cannot have any idea whether there was one or many objects along the path of the beam, so we cannot reconstruct the full image, however we begin by creating an image based on this information. The process of back-projecting a 1D signal across a 2D area is referred to as smearing. In this context it simply means duplicating the same 1D signal across all the columns. Then we can rotate the rays and send them at 90°. And get a similar, but vertical strip. We can repeat doing the same for several angles and later by adding those images we can get inverse transformation.

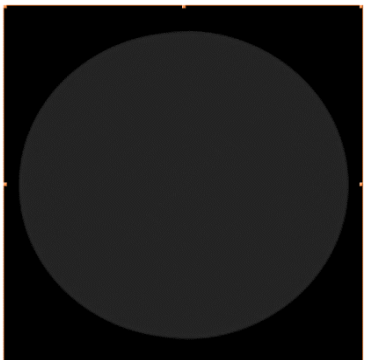


Fig.2. Inverse Radon of a circle using 360 back-projections.

Now, the grayscale image is an N x M matrix of pixel intensities. In this paper these pixel intensities are going to be viewed as absorption values at i, j^{th} pixel. We can then shoot a set of n parallel beams, henceforth referred to as rays, through the image and calculate the sum of pixel intensities. However, the rays might not pass through any pixel, but pass through a space in between. In that case we will interpolate the value at some position by using the absorption values of neighboring pixels.

The problem setting:Given a grayscale image calculate its Radon and inverse Radon transformations, extend the implementation to 3-channel RGB images.

It is assumed throughout this paper that the pixel intensities are given double precision floating point format in range [0,1]. In this paper we suggest a method of calculating the Radon Transform using bilinear interpolation. And inverse Radon transform using linear interpolation.

II. THEORY

2.1. Dirac Function

In order to understand how the Radon Transform works, firstly we need to understand what is the Delta function.Dirac's Delta Function is defined by:

$$\delta(t) = \begin{cases} 0 & t \neq 0 \\ \infty & t = 0 \end{cases} \quad (1)$$

This is an "Infinitely Peaked" function at $t = 0$ and the total area under$\delta(t)$ is 1. That is:

$$\int_{-\infty}^{\infty} \delta(t)dt = 1 \quad (2)$$

An important property of the Delta function is the following:

$$\int_{-\infty}^{\infty} f(t)\delta(t-k)dt = 1 \quad (3)$$

2.2. Continuous Radon Transform

Any straight line in the Cartesian plane $y = a \cdot x + b$ can be written in its normal form(see Figure 3.)

$$x \cdot \cos\theta + y \cdot \sin\theta = p$$

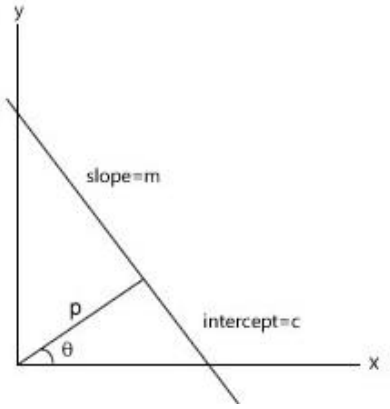


Fig.3.Normal form of a line.

Now we will define the Continuous Radon transformation as:

$$R(p,\theta) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\delta(x \cdot \cos\theta + y \cdot \sin\theta - p)dxdy \quad (4)$$

Where $\delta(\ )$ is the Dirac function. By applying the property in Equation 3 of the Dirac's Delta function to Equation 4 we get:

$$R(p,\theta) = \int_{-\infty}^{\infty} f\left(\frac{p - y * \sin\theta}{\cos\theta}, y\right) dy \quad (5)$$

2.3. Discrete Radon Transform

The Discrete Radon Transform on an $N \times M$ image $R(p,\theta)$ is defined as:

$$R(p,\theta) = \sum_{x=0}^{N-1}\sum_{y=0}^{M-1} f(x,y)\delta(x \cdot \cos\theta + y \cdot \sin\theta - p) \quad (6)$$

Where $N = M = 512$, $x,y \in [0,511]$, $\theta \in [0,\pi)$ are independent variables and $p = x \cdot \cos\theta + y \cdot \sin\theta$ is the dependent variable, $f(x,y)$ denotes the intensity of the pixel located in the (x,y) position of the original image. For any fixed θ, when p varies we calculate the intensity of the image along the line defined by θ and p This gives us one projection. If we repeat this process for various θ-s we will get different projections of the same image under different angles.

The Radon transform is sometimes displayed as an image with p and θ as it's rectilinear coordinates. This image is often refereed to as sinogram, because the Radon transform of an off-center point sourceis a sinusoid.

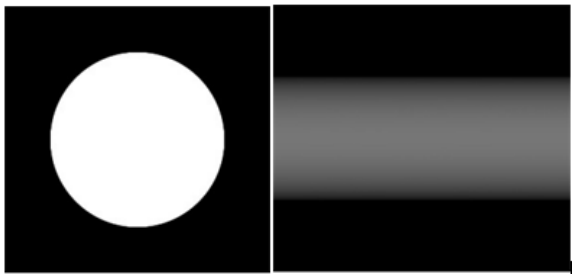


Fig.4. (a,b)Example of aSinogram.

In Figure 4 (b) is the sinogram of (a), that is, after theRadon transformation of the image (a),that is, applying Formula 6to the image, we get the image (b).

2.4. Inverse Discrete Radon Transform

The inverse Radon transform is widely used in image reconstruction, the sinogram (Radon transform of an image) contains the necessary data to reconstruct that image. The idea is to back-project all projections and then add them together to get the initial image.

For a fixed value of θ_k and p_j (a single point) the back-projection of that point is just a line on the image, such that, the value of each point on that line is $R(p_j,\theta_k)$. We repeat this process for all values of p_j keeping θ_k fixed to get the following result:

$$f_{\theta_k}(x,y) = R(p,\theta_k) = R(x \cdot \cos\theta_k + y \cdot \sin\theta_k, \theta_k) \quad (7)$$

The equation 7 holds for any value of θ_k, thus we can write that the image formed from a single back-projection for any angle θ is given by:

$$f_\theta(x,y) = R(x \cdot \cos\theta + y \cdot \sin\theta, \theta) \qquad (8)$$

To obtain the original image we need to add the back-projections for all angles. That is, we need to integrate over all the back-projected images, i.e.

$$f(x,y) = \sum_{\theta=0}^{\pi} f_\theta(x,y) = \\ = \sum_{\theta=0}^{\pi} R(x \cdot \cos\theta + y \cdot \sin\theta, \theta) \qquad (9)$$

For the continuous case Equation 9 is transformed into $f(x,y) = \int_0^\pi f_\theta(x,y)d$.

A back-projected image formed this way is sometimes referred to as a laminogram. A laminogram is only an approximation of the original image, it is not the exact image. For example, the laminogram of Figure 5 a is b:

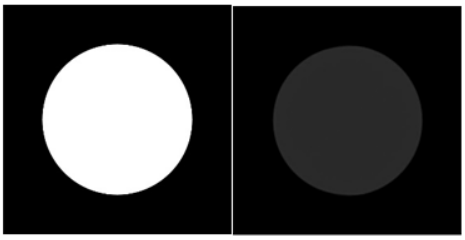


Fig.5. (a, b) Example of aLaminogram.

We can clearly see that the laminogram is blurry and darker than the original, but it is not a bad approximation.

III. NUMERICAL SOLUTION

3.1. Composite Trapezoid Rule

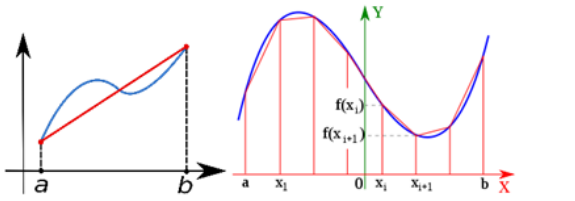
The Composite Trapezoid Rule/Method is a Quadrature Rule, i.e. it is a method for calculating definite integrals.

$$\int_a^b f(x)dx \qquad (10)$$

The Composite trapezoid rule works by dividing the interval $[a,b]$ into n small equal sub-intervals and then for every sub-interval $[x_i, x_{i+1}]$ joins the points $(x_i, f(x_i))$ and $(x_{i+1}, f(x_{i+1}))$ by a straight line and calculates the area of the trapezoid under that line(see Figure 6).

Here $x_i = a + i \cdot \frac{b-a}{n}$, $i = 0,1,\dots,n$. Also denote the mesh size by $h = \frac{b-a}{n}$. The approximate value of the integral will be:

$$\int_a^b f(x)dx \approx \frac{h}{2} \cdot \sum_{i=0}^{n-1} \left(f(x_i) + f(x_{i+1})\right) \qquad (11)$$



Trapezoid Rule (n=1) Composite Trapezoid Rule

Fig.6. (a, b) Numerical Integration.

3.2. Bilinear Interpolation

Bilinear interpolation is an extension of linear interpolation for interpolating functions of two variables on a 2D grid. Assume we need to find the value of the function f at the point (x,y) and we know the values of f at 4 points $a_{00} = (x_0, y_0)$, $a_{01} = (x_0, y_1)$, $a_{10} = (x_1, y_0)$ and $a_{11} = (x_1, y_1)$.

First, we calculate $f(x, x_0)$ and $f(x, x_1)$. By using simple linear interpolation in the x-direction we will get that:

$$f(x,y_0) \approx \frac{x_1 - x}{x_1 - x_0} \cdot f(a_{00}) + \\ + \frac{x - x_0}{x_1 - x_0} \cdot f(a_{10}) \qquad (12)$$

And, similarly:

$$f(x,y_1) \approx \frac{x_1 - x}{x_1 - x_0} \cdot f(a_{01}) + \\ + \frac{x - x_0}{x_1 - x_0} \cdot f(a_{11}) \qquad (13)$$

Then we need to calculate the value of $f(x,y)$ by linearly interpolating in the y-direction, and we get:

$$f(x,y_1) \approx \frac{y_1 - y}{y_1 - y_0} \cdot f(x,y_0) + \\ + \frac{y - y_0}{y_1 - y_0} \cdot f(x,y_1) \qquad (14)$$

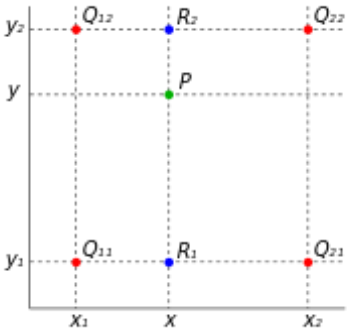


Fig.7.Bilinear interpolation to find the point$f(x,y) = P$.

3.3. Algorithm of Radon Transform

The problem statement: Given a grayscale 512×512 image I and an integer $0 < n < 2000$, compute its Radon Transform at n angles with 725 sensors equally spaced - outputting the sinogram of size (725xn).

We propose the following approach to implement the Radon Transform. First give the number n of angles α considered. That is, we do the transformation under n distinct angles. We calculate α_i for any $i = 0,1,2,\dots,n-1$ by the following formula:

$$\alpha_i = \frac{180 \cdot i}{n} \quad (15)$$

Now for every angle$\alpha_i$we need to integrate our image on unidistant (the proposed distance is 1) lines of degree α_i. First, we need to find the equation of those lines. To calculate those, we will use the normal form of the line, that is:

$$x \cdot \cos\alpha + y \cdot \sin\alpha = p \quad (16)$$

Integral is substituted by summation of pixel intensities, so we need to choose a frequency at which we are going to sample the image in the direction of each line. Experimental results show that $dp = 1$ is gives the best quality to running time ratio.

We take $725 = \sqrt{2} \cdot size$ (to cover the diagonal of the 512×512 square) lines,so the middle line, 363th line will definitely pass through the midpoint of the square, and from here we can find p_{363} for that line. Afterwards we calculate $p_1 = p_{363-362}$. Now we fill the sinogram:

$$sinogram[i][j] = integrate(\alpha_i, p_i) \quad (17)$$

Where integrate is a function, which, after calculating the intersection (by a function intercept) of the image's borders with the line (α_i, p_i) integrates over the region of intersection.

Intercept

The intercept function returns the coordinates (x, y) of the first intersection of the line with the square and the direction in which x and y need to propagate further, that is: dx and dy. We consider 2 cases:

First - if $\left(0 \le \alpha < \frac{\pi}{2}\right)$:

$$y = size = 512; \quad x = \frac{p - y \cdot \sin\alpha}{\cos\alpha}; \quad dx = dp \cdot \sin\alpha; \quad dy = -dp \cdot \cos\alpha \quad (18)$$

Check if the point (x, y) is within the borders of our square by a function incheck (which returns yes, if the points is within the borders and no otherwise). If the result is yes, then we found our point, otherwise we look for the intersection with another border of the square.

$$x = 0; \quad y = \frac{p}{\sin\alpha}; \quad dx = dp \cdot \sin\alpha; \quad dy = -dp \cdot \cos\alpha \quad (19)$$

And again, check if the point (x, y) is in the square if it is, then we have found our point, if not, then we take a point outside of the square, thus that point will be skipped when integrating.

Second – if$\frac{\pi}{2} \le \alpha < \pi$):

$$x = 0; \quad y = \frac{p}{\sin\alpha}; \quad dx = dp \cdot \sin\alpha; \quad dy = -dp \cdot \cos\alpha \quad (20)$$

Check if the point (x, y) is within the borders of our square. If the result is yes, then we found our point, otherwise we look for the intersection with another border of the square.

$$x = p/\cos\alpha; \quad y = 0; \quad dx = dp \cdot \sin\alpha; \quad dy = -dp \cdot \cos\alpha \quad (21)$$

Integrate

The function integrate takes as parameters an angle α and distance given initial value $sum = 0$. For the given line (α_i, p_i) we calculate the corresponding point of intersection (x_i, y_i), dx_i and dy_i.

While the point (x_i, y_i) is inside the borders of the square, or on the borders:

$$sum = sum + dp \cdot \left(\frac{f(x_i, y_i) + f(x_{i+1}, y_{i+1})}{2}\right) \quad (22)$$

and on each step, we increment x and y by:

$$x_i = x_i + dx_i, \quad y_i = y_i + dy_i \quad (23)$$

The function later returns sum, which is simply the value of the integral on the line (α_i, p_i).

Adjust Image

After the integrations - it is almost definite that the values of the sinogram are not in the range $[0,1]$. Our initial assumption was that we are dealing with Real representation of the image therefore we need a function to adjust the image to the range $[0,1]$. This is done in the following manner.

1. Find the minimum and maximum pixel intensities.
2. If the maximum-minimum is 0.0 exit the function.
3. Else - $image[x][y] = \frac{(image\ [x][y] - min\ )}{max}$

3.4. Algorithm of Inverse Radon Transform

The problem statement:Given a Sinogram of $725 \times n$ compute its inverse Radon Transform. Where n is the number of angles at which initial Radon transform was performed.

Linear interpolation

First, we will define a linear interpolation function which will be later used for the inverse Radon transformation.

This function takes 2 values t and an integer k. Then we initialize:

$$tl = \lfloor t \rfloor, \ th = tl + 1 \text{ and } dt = t - tl \quad (24)$$

The function returns:

$$sinogram[th][k] \cdot dt + sinogram[tl][k] \cdot (1 - dt) \quad (25)$$

Inverse_Transformation

We get each pixel at the position i, j in the following way:

For each integer $k \in [0, n)$, $\alpha = \pi \cdot \frac{k}{n}$

$$p = \left(i - \frac{512}{2}\right) \cdot \cos\alpha + \left(j - \frac{512}{2}\right) \cdot \sin\alpha + \left\lceil \frac{\sqrt{2} \cdot 512}{2} \right\rceil$$

$$sum = sum + linear_interpol(p, k)$$

$$laminogram[i][j] = sum \cdot d\alpha = sum \cdot \frac{\pi}{n}$$

After we do this for all $[i, j]$-s, where $i, j \in [0,511]$ and both i and j are integers, we will get the complete laminogram, i.e. the image after undergoing Radon transformation and inverse Radon transformation.

IV. RESULTS AND DISCUSSION

Let's look at some results obtained by our implementation of Radon Transform and Inverse Radon Transform.

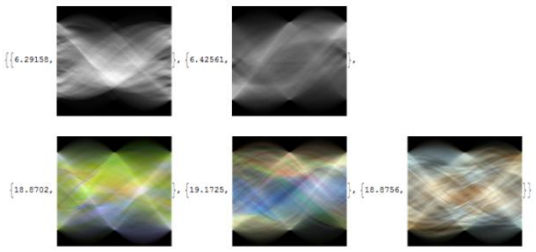


Fig.8.Radon Transform performed by our algorithm and corresponding running times (in seconds).

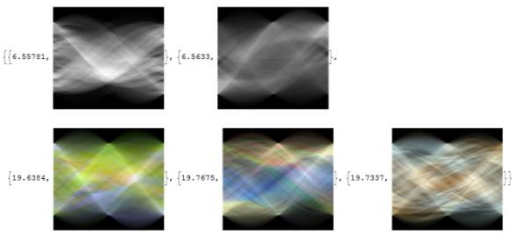


Fig.9.Radon Transform performed by our algorithm and corresponding running times (in seconds).



Fig.10.Radon Transform performed by our algorithm and corresponding running times (in seconds).



Fig.11.Radon Transform performed by our algorithm and corresponding running times (in seconds).

```
In[21]:= f[img_, img1_] := ImageDistance[img1, img, DistanceFunction
In[22]:= MapThread[f , {imageList, merInverse[[All, 2]]}]
Out[22]= {0.0676338, 0.0476131, 0.0475641, 0.0533084, 0.0197144}
In[23]:= MapThread[f , {imageList, irancInverse[[All, 2]]}]
Out[23]= {0.0669487, 0.116204, 0.0780402, 0.127277, 0.090136}
```

Fig.12.Earth Mover's Distance between inverse RadonTransforms and originalimages, first output is the differencebetween our inverse Radon transform andoriginal images.

Automotive engines and diesel equipment. The use of TBC on diesel components such as valves, pistons and fire decks insulates the metal substrates from high-temperature oxidation and corrosive environments. As a thermal barrier, it reduces metal temperatures as well as bond coat. The micro-columnar grains encircled in Fig.2. was observed, which were solidified from the melted fraction of YSZ

V. CONCLUSIONS

In this paper we have designed a method to implement Radon and Inverse Radon transformations. We have run our program and algorithm implemented in Wolfram Mathematica with ramp filter on images with different contrasts, colors and number of channels. Qualitatively we can see that our results with the test images overall give better contrast.
To emphasize the difference let's look at some quantitative comparisons of Statistics:
Firstly, let's take a set of 5 images - 2 grayscaled, 1 Jpg, 2 Png full color images and perform Radon and inverse Radon transforms - comparing the results to the corresponding original images.

We are going to use ImageDistance with a distance function of EarthMoverDistance (which isn't explained in the scopes of this paper). Overall the Earth Mover's Distance gives the minimum cost of turning one image into another. What the function returns is basically a number in [0,1] showing the dissimilarity of the two input images. As we can see our qualitative judgement is also backed up by some quantitative evidence.

Another important aspect is the running time of the functions:
As seen in our statistical experiments - our Radon transform works approximately with the same absolute time as Wolfram Mathematica's Radon function. Our Inverse Radon is 2-3 times slower than Wolfram Mathematica's implementation. This can be due to the fact that Mathematica uses paralleled implementations.

REFERENCES

[1] Holschneider M. Inverse Radon transforms through inverse wavelet transforms. Inverse problems. 1991 Dec;7(6):853.

[2] Durrani, T. S., and D. Bisset. "The Radon transform and its properties." Geophysics 49, no. 8 (1984): 1180-1187.
[3] Bracewell, Ronald Newbold, and Ronald N. Bracewell. The Fourier transform and its applications. Vol. 31999. New York: McGraw-Hill, 1986.
[4] Beylkin, Gregory. "Discrete radon transform." IEEE transactions on acoustics, speech, and signal processing 35, no. 2 (1987): 162-172.
[5] Ostrowski, Alexander M. Solution of equations and systems of equations: Pure and applied mathematics: A Series of monographs and textbooks. Vol. 9. Elsevier, 2016.

★★★