DICTA2002: Digital Image Computing Techniques and Applications, 21-22 January 2002, Melbourne, Australia.

Image Operations in Discrete Radon Space

Imants Svalbe
School of Physics and Materials Engineering
Monash University, 3800, Australia
imants.svalbe@spme.monash.edu.au

Abstract

The Discrete Radon Transform (DRT) preserves discrete digital image information whilst recasting 2D image data in a form that closely resembles 1D analog projections. The projective representation makes the DRT an effective tool for data compression and for tomographic reconstruction, in particular to obtain images from a limited set of real projection data. For prime sized images formed on regular square or hexagonal arrays, the projective mapping operation is arithmetic addition. Algorithms are presented for efficient computation of the forward and inverse DRT transformation and to perform elementary image rotation and translation operations in the digital projection space. Manipulating image data as DRT projections rather than in the spatial domain avoids the need for expensive reconstruction and re-projection of the data. Comparing objects in projection space may accelerate iterative reconstruction schemes. For some general image-processing operations, projection-based algorithms may produce efficiencies not realisable in the spatial domain. The patterns of pixel locations that combine to form each projection element are shown to have interesting distributional properties that derive from the prime, cyclic nature of the DRT.

Key Words: *discrete image processing, computer algorithms, Radon transforms.*

1. Introduction

Tomography plays an important role in fields ranging from medical and industrial imaging to aperture synthesis radioastronomy and, in the earth sciences, seismic profiling. The interplay between the process of collecting discrete signal samples, the formation of object projections in continuous (Radon) space and the creation of a digital reconstructed image has been of considerable interest for over 20 years [10].

A very general definition for a Discrete Radon Transform (DRT) was developed by Beylkin [1] for discrete arrays of arbitrary integer size. Beylkin presented an exact, algebraic formalism for the discrete projection and inversion process, connected the DRT with the discrete Fourier transform and included the possibility of the projections being taken along curved as well as straight paths. Beylkin also foreshadowed the implementation of filtering and other spatial operations in 2D or higher spaces by computation of these properties in the DRT space. That topic forms the theme of this paper.

The digital projections used here are those defined in Matus and Flusser [2]. They outlined the group theoretic and Fourier interpretations of the DRT as applied to square, prime-sized arrays. They described an application of the DRT as a mechanism for relatively efficient data compression in digital images. When the array size is restricted to be a prime rather than arbitrary integer, the pattern of sampled image data for each transform point is unique for all projections; this decouples the line integrals. The projection and inversion process then becomes a simple arithmetic (additive) operation rather than more general algebraic transformation of Beylkin's DRT scheme.

Salzberg [3] applied the DRT to image the structure of 3D crystals from their real x-ray transmission profiles and showed it was possible to invert real x-ray projection data using the DRT under an iterative reconstruction formula. Svalbe developed a general algorithm [4] that applies the DRT formalism to directly compute the grey scale image from real space projection data. That paper shows that real projection data can be mapped into a discrete form compatible with the DRT. This allows the DRT to be used for real to discrete data mapping, as well as for discrete to discrete mapping. The DRT has other varied applications. In [5], the DRT is used for the imperceptible, orientation-sensitive embedding of messages or structural information in discrete images. In [6], the distribution of gap lengths between digital samples along digital projection rays of the DRT is shown to be related to the chaotic energy spectra observed in discrete systems. The prime length DRT has links through this to the distribution of zeros of the Riemann Zeta function.

A “grid-friendly” mapping that is different to the DRT but that that also preserves discrete line structures was

developed recently [10], to allow exact mapping between digital lines and their Radon transform on arbitrary discrete arrays. This mapping has the representational advantage that individual lines do not wrap around the digital array, as occurs in the DRT. The inverse mapping, although iterative, converges rapidly and accurately to the digital lines as input.

Here we present algorithms to compute the forward and inverse DRT. We also examine methods to perform elementary image operations such as image rotation and translation in the digital projection space. The object of these operations is to perform image comparisons and filtering directly on the projection data without requiring reconstruction of the image. For iterative algorithms, this may result in substantial computational savings, as well as providing alternative methods for image processing operations that are usually performed in the spatial domain.

The DRT R(t, m) of image I(x, y), is defined [2, 4] by the parameters p, t and m, where p is the prime array size of the (square) 2D image data. The integer index m, with $0 \le m \le p$, fixes (uniquely) the relative offsets, x_m:y_m, between nearest neighbour pixels of the digital ray for that m value. The ray angle, $\theta_m = \text{atan}(y_m/x_m)$, corresponds to the orientation of the line joining pixel positions selected by (t, m) to their nearest neighbours. The minimum distance between neighbouring pixels is d_m, where $d_m^2 = x_m^2 + y_m^2$. The value of x_m is allowed to take negative values, to enable the digital projections to span angles from 0° to 180°. This leads to fourfold and sixfold symmetry in the angle and gap distributions, for the square and hexagonal lattices respectively. For the hexagonal array, x_m and y_m are both odd or even.

The integer index, t, $0 \le t < p$, measures the translation of a digital ray in horizontal increments from t = 0 at the top left corner of the image (where x = y = 0). An image point (x, y) maps [6] to translate t for a given m on a regular square lattice according to

$$t = (x - my) \bmod(p)$$

$$m = (\alpha p + x_m)/y_m \text{ , and} \qquad (1)$$

$$t = (x - (m + \tfrac{1}{2})y) \bmod(p)$$

$$m = [(\alpha p + (x_m - y_m)/2]/y_m \qquad (2)$$

for regular hexagonal lattices. Modulus p arithmetic applies in (1) and (2); α is the smallest positive integer for m to be an integer.

2. Forward and Inverse DRT Algorithms

The algorithms for digital projection and image reconstruction [2] are remarkably simple, but can be improved by exploiting the uniform definition of translation that applies for all projection angles [4]. Equations (1) and (2) show that unit translate increments of t in R(t, m) correspond to unit increments of x in I(x, y). At a given m, rows of R can be constructed by adding to R a block copy of the data on each row of I(x, y). The location $x = x_0$ at which t = 0, shifts by m mod(p) as y increments. The "block copy and add" from I into a complete row of R is done in two contiguous parts, $x = x_0$ to p - 1 and x = 0 to $x_0 - 1$. The block copy operation is a frequently used and hence relatively efficient computer instruction for most CPU architectures, as it accesses and processes a string of data items that are stored sequentially in computer memory.

The typical computation time required to construct a 761 x 762 projection from 761 x 761 floating point image data decreased by a factor of ≈ 5; taking 221.81 seconds for the standard algorithm [2] and 40.78 seconds for the block copy method. Times are cited for a single CPU serial computer. For 479 x 479 data, the times were 43.33 and 9.61 seconds respectively.

The usual inverse DRT algorithm [2] writes back the contents of each element of R(t, m) into the reconstructed image, I'(x, y), at the same pixel locations as those accessed to sum the values for that digital projection. The forward projective transformation applied to the first p rows of R will also produce a reconstructed image. The effective value of m, m', for the re-projected data can be obtained from (1) by subtracting the translates produced for adjacent rows in R, as

$$m' = t'_{i+1} - t'_{i+1} = t - (i + 1)m - (t - im)$$
$$= - m = p - m$$

so that m' is the complement of m.

Double projection maps I(x, y) through R(t, m) to I'(x, p-y) so that $R^2 I \cong -I$. The final row in R, m = p, must be back projected separately (this is a consequence of an arbitrary decision to map rows to an angle of 90° and columns to 0°). The projected values are normalised by subtracting the total intensity of any one row of R(t, m) (equivalent to the total image brightness) and then dividing by p. I'(x, y) then has exactly the same image values as the original image I(x, y).

Using the forward transform to perform the inverse mapping is an advantage if the read time per pixel is less

than the write time per pixel for a given computing system. Parallel computers can more easily read multiple data items than write them, as this causes fewer memory contention problems. Reconstructing the image using the forward transform reads the data p times from R for each (t, m), with the summed result being written once into the final image I'(x, y). The conventional inversion algorithm reads once from R and writes this result p times into I' for each element (t, m). For the "block copy and add" approach, the relative gains in processing speed are smaller, as adjacent translates are still read and added (hence written) to the output in sequential translate order (processing all t for each m, rather than all m for each t).

The typical computation time to reconstruct a 761 x 761 image from 761 x 762 floating point R data was 43.30 seconds using the inverse transform and 40.82 seconds for the forward transform inversion method, on a single CPU serial computer. Both approaches used the more efficient "block copy and add" approach.

For each row increment, t must be incremented by m, then t is checked for $t \geq p$ (with p subtracted if required). This operation is done p + 1 times for each DRT projection. Pre-computing a list of the wrap offsets and then indexing into this array for each row might be thought to give some marginal gain, but generated at best equal or larger computational overheads (taking 41.35 seconds for image reconstruction from 761 x 762 data).

The approaches considered here are all variants of the standard back projection algorithm (and scale as order n^3; n backprojections are performed, each over an n x n image). Faster projection and inversion algorithms may be achieved using Fourier or the related linogram (order $n^2 \log n$) approaches. The latter methods both require re-sampling of the I(x, y) or R(t, m) values. Here we have concentrated on exact inversion methods using only addition of the original sample values, so that the errors in I'(x, y) – I(x, y) are limited only by the finite precision with which the (floating-point) data values are represented, and to avoid the iteration required for inversion methods such as [10].

3. Image Translation and the DRT

To translate an object by an offset of dx in the x direction and dy in the y direction in continuous projection space, (ρ, θ), requires angle dependent interpolation and resampling of the projections, as $\rho' = (x - dx)\cos\theta + (y - dy)\sin\theta$. A translation operator T can be applied easily in the R domain because of the linear and uniform definition for translates t within R, being the same for each m projection.

For the square lattice, the required offset of the translate origin, dt, depends linearly on the row index m and is given, from equation (1), by $dt = -dx + mdy$. The location of t = 0 is moved by an integer offset, dt mod (p), with the shifted data from row m of R being copied to form the translated projections for row m of R'. This block copy is easily done in two parts, t = dt to t = p – 1 and then t = 0 to t = dt –1, as used in the forward and inversion projection algorithms, for each m. For m = 0, dt = p – dx and for m = p, dt = p – dy.

For hexagonal offsets, equation (2) gives the translate mapping, T, as $dt = -dx - m(p - dy) + (dy + 1)/2$, with dt = -dy for m = 0 and $dt = -dx + p + dy/2$ for m = p. If R(t, m) is the DRT of the original image, then the DRT of the translated image is given by R'(t', m) = TR(t, m).

Figure 1. Reconstructed image after translation by offsets dx = 100 and dy = 200 in R(t, m) space (479 x 479 image).

Translates wrap in unit increments around hexagonal arrays in the x direction, but in the y direction the distance between vertically adjacent pixels is 2 units (of size $\sqrt{3}/2$ as adjacent rows have pixels offset by ½ a translate unit). The above mapping preserves exactly the spatial relationships of all pixels from p – dy to dy. Those pixels wrapped by the translation process vertically around the array have an alternating unit horizontal offset with opposite phase to that of the p x p lattice. To undo the translation, operator T' may be applied with offsets dx' and dy' so that dx' + dx = 0 and dy' + dy = 0. To correctly

account for the effects of the vertical wrapping, the negative offsets are dx' = (p + 1)/2 - dx and dy' = p – dy. Then applying $T^{-1}TR = R$ exactly.

Figure 1 shows an example image reconstructed after a translation by dx = 100 and dy = 200 in the R(t, m) space. The original image was a 479 x 479 image ("Lena") mapped to a hexagonal lattice.

4. Image Rotation and the DRT

Rotation of an object by an angle dθ about the image circle centre in (ρ, θ) space can be done trivially by redefining the row of sinogram data that corresponds to θ = 0. If the angle dθ does not point to one of the existing projection angles, then a new set of projections need to be re-interpolated from the existing set. For discrete images in 2D space, general rotations require interpolation of the data, for which many efficient algorithms [7] are available.

4.1. Rotation by 90° on the Square Array

After a 90 degree rotation, translations are effectively taken along the image column directions. It is possible to map the (t, m) coordinates of a projected image into those of the 90 degree rotated image (t', m'), that is R(t', m') = SR(t, m), where S is the 90 degree rotation operator.

The mapping S between values m and m' is generated by forming the sequence seq[m'] = (αp – 1)/m. As rows and columns are interchanged, seq[0] = p and seq[p] = 0. This transform makes $\theta_m' = 90° + \theta_m$ by mapping x_m:y_m to – y_m:x_m. Then, from equation (1), $m' = (\alpha p - y_m)/x_m$. Multiplying top and bottom by $1/y_m$ gives m' = (α'p – 1)/m, using modulus arithmetic, where α' is some other integer. For example, when p = 7, seq[m'] = 7, 6, 3, 2, 5, 4, 1, 0. Figure 2 shows the mapping on R produced by S for clockwise and anticlockwise 90° rotations on a 7x7 image.

00 10 20 30 40 50 60	**07 17 27 37 47 57 67**	**67 57 47 37 27 17 07**
01 11 21 31 41 51 61	**66 06 16 26 36 46 56**	**66 56 46 36 26 16 06**
02 12 22 32 42 52 62	**63 33 03 43 13 53 23**	**33 63 23 53 13 43 03**
03 13 23 33 43 53 63	**62 42 22 02 52 32 12**	**22 42 62 12 32 52 02**
04 14 24 34 44 54 64	**65 15 35 55 05 25 45**	**54 34 14 64 44 24 04**
05 15 25 35 45 55 65	**64 24 54 14 44 04 34**	**45 15 55 25 65 35 05**
06 16 26 36 46 56 66	**61 51 41 31 21 11 01**	**11 21 31 41 51 61 01**
07 17 27 37 47 57 67	**60 50 40 30 20 10 00**	**00 10 20 30 40 50 60**
(a)	(b)	(c)

Figure 2. a) (t, m) coordinates for a 7x7 image, b) the DRT coordinates (t', m') for the image rotated 90° anticlockwise by SR, c) the DRT coordinates (t', m') for the 90° clockwise image rotation, by $S^{-1}R$

The mapping of translates from R(t, m) into R'(t', m') is also cyclic in m'. Figure 3 shows that the location of samples in I(x, y) that are spaced a horizontal distance of m units apart in R space (ie at translations t mod(m)) will always correspond to vertically adjacent elements in I(x, y). For the example shown in Figure 2, at m' = 2, the translates from m = seq[2] = 3 are mapped linearly across the row at every second translate as [– 0 – 1 – 2 – 3 – 4 –5 – 6], which becomes, for mod(7) lengths and starting at t = 2, the seven translates [6 3 0 4 1 5 2] (as observed on the third row of Fig. 2b, where m' = 2).

o a o o o o b o o o o c o o o o d o o o o e o o o …..
o a o o o o b o o o o c o o o o d o o o …..
o a o o o o b o o o o c o o o …..
o a o o o o b o o o …..
o a o o o …..

Figure 3. The top left corner of an arbitrary image, I(x, y). Image samples separated by a horizontal distance of m in I(x, y), marked by the values a, b, c, d, and e, represent vertically adjacent elements in the DRT space R(t, m). Here t = 1 and m = 5, "o" marks the location of other image pixels.

As successive applications of S form cumulative rotations, SSSSR = S^4R = R. Permutation of the steps outlined above produces a single pass algorithm equivalent to 180 degree rotation, S^2, and yields another for 270 degree rotation, $S^3 = S^{-1}$, so that $S^{-1}SR = R$. The mapping seq[m] is related to the generalised Legendre sequences generated by the powers of prime residues used in active beam steering [8].

4.2 Rotation by 60° on the Hexagonal Array

The 60 degree rotation operator, H, is the transform that makes $\theta_m' = 60° + \theta_m$. The same formalism mapping m to m' and t to t' applies to H as for S, except that here seq[m] is one less than for S (the complementary angle to m is p – 1 – m for hexagonal arrays [6]). The special case m = 0 is mapped to m' = p, m = p is mapped to m' = p – 1 and m = p – 1 is mapped to m' = 0. The latter two translation assignments need to be reflected to preserve the left to right definition of translation adopted here. The operation $H^3R = H^{-1}HR = R$.

The operators S and H represent 2D computations in I(x, y), but are 1D computations when performed in R(t, m). Rotations in R are generally faster to implement than the equivalent rotations in I, as the data access overhead for row based operations is much smaller and the efficiency of data caching is likely to be higher. For a 761 x 761 image, an efficient row/column swap took 0.27

seconds, in R space, the same operation took 0.15 seconds. Figure 4 shows the reconstructed image resulting from $R^{-1}HR$ where R is the DRT of the original version of the image shown in Figure 1.

Figure 4. Reconstructed image after the 60° rotation operator, H, was applied to the DRT of the original of the image shown in Figure 1 (479 x 479 image).

4.3 Combinations of Rotation Operations

Combinations of translation and rotation operators in R can be cascaded to produce offset and rotated images. Translations are commutative with rotations as they permute only the order of translations on each row of R, for example TSR = STR. Cyclic warps of the image can be achieved by cascading the operators T, S and H.

Figure 5 shows the reconstructed image obtained after the hexagonal rotation operator, H, is applied to the square lattice digital projections, producing a 45° warp of the original data.

The operators H, S and T shuffle the coordinates of the original image DRT. Rotations at arbitrary angles and warp factors will require interpolation of the DRT values. Elementary operators such as S and H exploit the original four or six fold angular symmetry of the set of digital projections. The definition of m = 0 and m = p that was adopted in [4] and is used here, requires that the content of the rows 0 and p in R are swapped each time the H and S operators are mixed.

Figure 5. Reconstructed image after the hexagonal lattice rotation operation H was applied to the square lattice DRT of the original image of Figure 1.

5. Properties of the DRT Basis Functions

The DRT mapping has a number of intriguing properties, beyond the number theory links described in [6]. The relationship between the m value which has the maximum value of d_m and d_1, the pixel gap distance for rays at m = 1 for a given p can be written as $d_1^2 = m^2 + m + 1 = \alpha p$, where α is an integer. This implies that an area equal to d_1^2 should contain $\alpha \bmod(p)$ pixels that belong to the digital rays at that m, for any t, for a given value of p.

Figure 6 shows the set of pixel locations in I(x, y) selected by the digital projection t = 0, m = 97 (where d_m has its maximum value), for p = 409. A square window of side length d_1 translated across this image does usually contain $\alpha \bmod(p) = 23$ pixels, because as one pixel leaves a translated version of the window, another enters at the opposite side of the window. This sliding window property is useful in creating arrays where each location has a requirement that a constant number of sources are located within a square region of a given size for any location within the (cyclic) array.

For 100,000 (uniformly) random placements of boxes of size 97 x 97, we find that on 99,497 occasions there were 23 pixels inside the box, whilst the remaining 503 occurrences had 24.

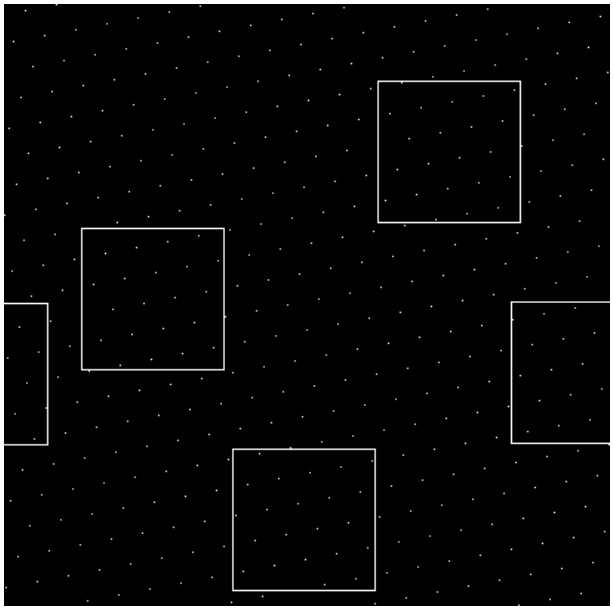

Figure 6. Dots are the pixel locations sampled by the DRT for p = 409, m = 97 and t = 35. The four squares drawn in the array are of size 97 x 97. Such squares enclose 23 pixels at almost all locations in the p x p array.

6. Conclusions

Algorithms that operate on, or permute the contents of discrete DRT projections have been presented. The resulting operations are efficient means of forward and inverse DRT mapping and to perform elementary translations (T) and rotations (S and H) from within the digital projection space. We are continuing to examine more general rotation operators, convolution and morphologic operations taken along periodic or connected lines [9] in the R arrays.

Figure 7 shows the point spread function resulting from convolution in R space along hexagonal axes. The operation was performed by uniformly weighted averaging using a length of 45 pixels) of the translates in the R space at three 60 degrees rotations. Wrapping across the t = 0 and t = p – 1 boundaries is accommodated by using circular convolution.

If G is a general linear convolution operator, then $GI(x, y) = R^{-1}G'RI(x, y)$. The convolution G' may be implemented in R space as the sum of translated DRT's over the kernel of the operator, as sums of y-translated 1D row convolutions or, for separable kernels, as the cumulative product of 1D convolutions over appropriate rotations of the image.

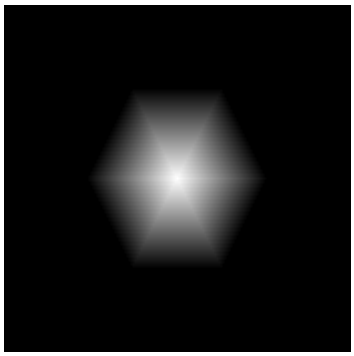

Figure 7. 173x173 image. Point spread function for an impulse at (86, 86) reconstructed after cumulative 1D convolution with a pixel length of 45, performed in R space at rotations 0, 60 and 120 degrees.

7. References

[1] G. Beylkin, "Discrete Radon Transform", IEEE Trans. on Acoustics, Speech and Signal Processing, vol. ASSP-35, no. 2, pp. 162-172, February 1987.

[2] F. Matus and J. Flusser, "Image Representation via a Finite Radon Transform", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 15, no. 10, 1993, pp. 996-1106.

[3] P. M. Salzberg and R. Figueroa, Chapter 19, "Tomography on the 3D-Torus and Crystals", in *Discrete Tomography: Foundations, Algorithms and Applications*, Eds. G. T. Herman and A. Kuba, Birkhauser, Boston, 1999.

[4] I. Svalbe and D. van der Spek, "Reconstruction of Tomographic Images Using Analog Projections and the Digital Radon Transform", presented at the Workshop on Discrete Tomography", Sienna, Italy, October, 2000 (in press, *Linear Algebra and its Applications, 2001*).

[5] I. Svalbe, "A Method to Embed Messages in Images Using the Digital Radon Transform", accepted for presentation at ICIP, Thessalonika, Greece, October, 2001.

[6] I. Svalbe, "Digital Projections in Prime and Composite Arrays", IWCIA, Philadelphia, August, 2001, also see Electronic Notes in Theoretical Computer Science, www.elsevier/.nl/locate/entcs.

[7] J. Foley et al, *Computer Graphics, Principles and Practice*, 2nd Edition, Addisson-Wesley, 1996.

[8] M. Schroeder, *Number Theory in Science and Communications*, Springer-Verlag, 3rd Edition, 1997.

[9] R. Jones and P. Soille, "Periodic Lines and Their Application to Granulometries", *Mathematical Morphology and its Applications to Image and Signal Processing*, Eds P. Maragos, R. Schafer and M. Butt, Kluwer Academic Press, 1996, p. 263- 272.

[10] A. Averbuch, R. R. Coifman, D. L. Donoho, M. Israeli and J. Walden, "Fast Slant Stack: A Notion of Radon Transform for Data in a Cartesian Grid which is Rapidly Computable, Algebraically Exact, Geometrically Faithful and Invertible", May 2001, TR #2001-11, www-stat.stanford.edu/research.