Fast Linear Hough Transform

Jean E. Vuillemin
Digital Equipment Corporation, Paris Research Laboratory
85 Av. Victor Hugo, 92563 Rueil Malmaison, Cedex France.

Abstract

The Hough Transform *is the choice technique for identifying straight lines through digital images, with applications to* high energy physics *and* computer vision.

Classical methods for implementing the Hough transform of a $N \times N$ binary image require to compute N^3 additions over $n = log_2(N)$ bits integers, hence nN^3 bit operations *per transform.*

We introduce a new algorithm for computing the fast Hough transform *FHT which only requires $log_2(N) \times N^2$ additions, for a total of n^2N^2* bit operations *per transform. The method is based on a* recursive *algorithm for raster-scan line drawing, which is different from Bresenham's* iterative *one [1].*

The FHT has a divide and conquer *recursive structure similar to that of the classical* fast Fourier transform *FFT algorithm, with* simpler *atomic operations - additions over n bits numbers - and a more complex interconnect. The FHT algorithm can readily be implemented in* software*. It maps into* hardware *as well, and we detail the structure of a bit-serial circuit for computing the FHT.*

1: Motivation

The Hough transform has proved useful in a variety of domains. Typical applications are found in computer vision, see [8], [5] or [6]. This work is motivated by one use of the Hough transform, in current high energy physics.

The *transition radiation detector* TRD is a problem in a series of benchmarks put forward by CERN[1] in [2]. The goal is to measure the performance of various computer architectures, in order to build the electronics required by the *Large Hadron Collider* LHC, before the turn of this century.

The function of the TRD is to identify the slope and intercept (entry point) of the most likely *straight line* particle trajectory through the digitized rectangular *region of interest* RoI. The RoI is a $W \times H$ boolean matrix, with W=96, H=32 for the TRD. Collisions within the LHC occur at the rate of 100KHz, namely each 10μsec.

We may assume that line trajectories of interest enter the RoI through a specific face - west - and that their slopes are all less than 45 degrees, in absolute value. This is achieved through a *linear time* pre-processing of the input, based on plane symmetries - after which particles traverse the RoI from left to right.

Thanks to the design of the analog to digital initial AD converters for the LHC, the TRD trajectories enter the RoI from the west face, and exit through the east. In a 96×32 grid, this further limits the slopes to 30 degrees, at most. By *re-sampling* the TRD input, horizontally within a 3 to 1 ratio, we reduce the rectangular problem 96×32 to a square 32×32 one, with trajectories under 45 degrees; input pixels now have two bits integer values. Without loss of generality, we thus

[1] Centre pour l'Etude des Réactions Nucléaires, in Geneva, Switzerland

restrict our attention to a square $N \times N$ RoI, with input pixels coded by $m \geq 1$ bits and $N = 2^n$ a power of two.

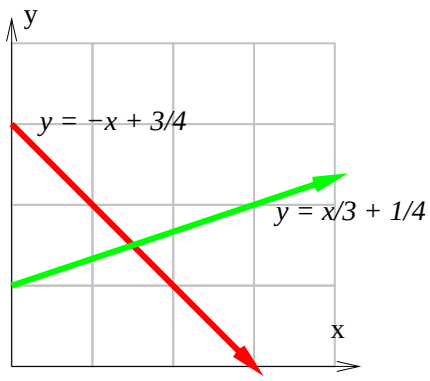
Previously reported implementations of the Hough transform all require to sum N numbers of $m+n$ bits, for each of the $(2N-1)N$ lines through the RoI. The total complexity is $2(n+m)N^3$ *bit operations* per transform. In section 2, we review such *naive* implementations; we summarize in section 3 the properties of the Hough transform, and in section 4 those of *digital lines*, which are relevant here.

We introduce in section 5, an algorithm for the *fast* Hough transform FHT, which reduces the complexity to $2n(n+m)N^2$ bit operations. The FHT speeds up *software* implementations of the TRD by a factor N/n; alternatively, the FHT reduces the amount of *hardware* required for implementing the FHT in real time - 100KHz for the TRD - by the same factor - over four times less gates and wires for the TRD, than with the naive method.

2: Linear Radon and Hough transforms

The (discrete) Hough transform is the limiting case of the (continuous) *linear Radon transform*, for zero width pixels [7], [4], [6].

In order to define the Radon transform, consider a two dimensional *continuous* image P, defined by some density distribution $P(x,y) \in \mathbf{R} \geq 0$, at each point $(x,y) \in \mathbf{R}^2$ within the Euclidean plane.



We use the above input RoI as a recurrent working example: domain P is composed of the two Dirac lines $y = -x + \frac{3}{4}$ and $y = \frac{x}{3} + \frac{1}{4}$.

In order to define the Hough transform, consider a two dimensional *discrete* image $\mathcal{P}_n$, defined for each integer pixel $(x,y) \in [0..N-1]^2$ within the square of side $N = 2^n$, by some m bit natural number $\mathcal{P}(x,y) \in \mathbf{N}$. One discrete RoI, corresponding to our working example, is:

$$\mathcal{P}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Let $D(s,i)$ represent the *continuous* straight line with slope s and intercept i, given by the equation

$$y = sx + i.$$

Let $\mathcal{D}_n(s,i)$ represent the *digital* line segment formed by pixels joining the grid point $(0,i)$ to $(N-1, i+h)$, with $N = 2^n$. One possible formula, defining $\mathcal{D}_n(s,i)$ for integer $0 \leq x < N$ and $0 \leq s \leq 1$, is the equation

$$y = \lfloor sx + i + \frac{1}{2} \rfloor.$$

Here, $\lfloor r \rfloor \in \mathbf{Z}$ is the *floor* of real $r \in \mathbf{R}$: $\lfloor r \rfloor \leq r < 1 + \lfloor r \rfloor$. It follows that $\lfloor r + 1/2 \rfloor$ is the nearest integer to the real r.

The *Radon transform* $R = \mathcal{R}(P)$ is the integral of all the points in P located on each continuous line D:

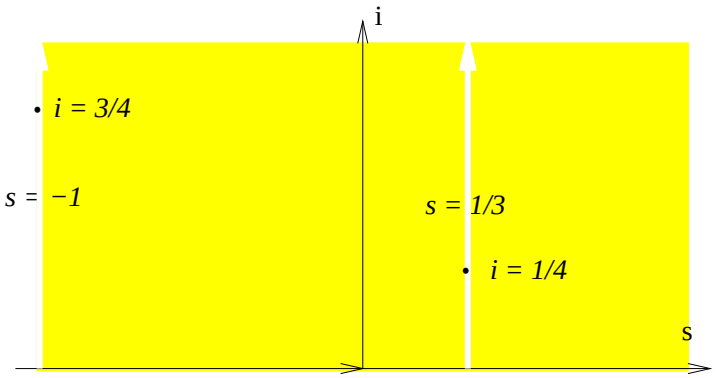
$$R(s,i) \quad = \quad \int_{-\infty}^{+\infty} dx\, P(x, sx + i). \tag{1}$$

The Radon transform $R(s,i) = \mathcal{R}(P)$, in our working example, is:

1 for $s = -1, i \neq \frac{3}{4}$ and $s = \frac{1}{3}, i \neq 1$;

2 for $s \neq -1$ and $s \neq \frac{1}{3}$;

∞ for $s = -1, i = \frac{3}{4}$ and $s = \frac{1}{3}, i = \frac{1}{4}$.



The *Hough transform* $H = \mathcal{H}(\mathcal{P})$ is the sum of all the points in $\mathcal{P}$ located on each discrete line $\mathcal{D}$:

$$H(s,i) \quad = \quad \sum_{0 \leq x < N} \mathcal{P}(x, \lfloor sx + i + \frac{1}{2} \rfloor). \tag{2}$$

The Hough transform of a square domain $\mathcal{P}_n \in N^2$ is a rectangular $(2N-1) \times N$ array $H_n(s,i)$, defined for integer slopes $-N < s < N$ and intercepts $0 \leq i < N$.

$$H_2 = \begin{bmatrix} 4 & 3 & 3 & 1 & 1 & 1 & 1 \\ 1 & 3 & 2 & 3 & 1 & 0 & 0 \\ 1 & 1 & 3 & 3 & 4 & 3 & 2 \\ 0 & 0 & 0 & 1 & 1 & 3 & 2 \end{bmatrix}$$

Definition (2) directly leads to an algorithm which computes the Hough transform, with $2N(N^2 - 1)$ additions. For each slope $|s| < N$ and intercept $0 \leq i < N$ in the transform domain, the *naive* method simply sums the N terms of (2), using $n + m$ bits of precision for the results.

3: Properties of the Hough transform

General properties of the Hough transform can be found in [6]. Let us summarize the ones which are relevant, both in the continuous and discrete cases.

We let $\delta(x)$ denote Dirac's point distribution, with its usual properties: $\delta(x) = 0$ for $x \neq 0$, $\delta(0) = \infty$ and $\int_{-\infty}^{+\infty} dx\, \delta(x) = 1$. We let ∂_a^b denote Kronecker's symbol: $\partial_a^a = 1$ and $\partial_a^b = 0$ for $a \neq b$.

(L) Both transforms are *linear*:
$\mathcal{R}(P+Q) = \mathcal{R}(P) + \mathcal{R}(Q)$ and $\mathcal{H}(P+Q) = \mathcal{H}(P) + \mathcal{H}(Q)$.

(P) The Radon transform of the Dirac point $\delta(|x-w| + |y-h|)$ is a line such that:

R(s,i)=1 when $h = sw + i$,

R(s,i)=0 otherwise.

The Hough transform of a digital point $\partial_w^x \partial_h^y$ is a digital line such that:

H(s,i)=1 for $h = \lfloor sw + i + \frac{1}{2} \rfloor$,

H(s,i)=0 otherwise.

(L) The Radon transform of a Dirac line $\delta(y - s'x - i')$ is:

R(s,i)=0 for lines which are *parallel* to $y = s'x + i'$, i.e. $s = s'$ and $i \neq i'$.

R(s,i)=1 for lines which *intersect* $y = s'x + i'$, i.e. $s \neq s'$.

R(s,i)=∞ for *the* line $y = s'x + i'$, i.e. $s = s'$ and $i = i'$.

The Hough transform of a digital line $\partial^y_{s'x+i'}$ is:

H(s,i)=0 for digital lines which are *parallel* to $\partial^y_{s'x+i'}$, i.e. $s = s'$ and $i \neq i'$.

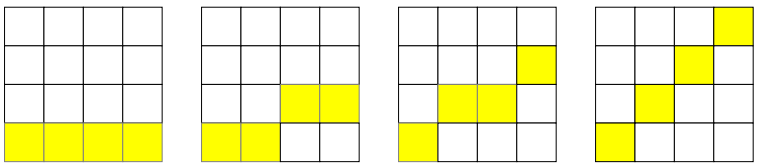
1≤H(s,i)≤ N/2 for digital lines which *intersect* $\partial^y_{s'x+i'}$, i.e. $s \neq s'$.

H(s,i)=N for *the* digital line $\partial^y_{s'x+i'}$, i.e. $s = s'$ and $i = i'$.

From (1), we see that the Radon transform of a finite set of $k \in \mathbf{N}$ lines is *infinite* at the corresponding k points in the transform space, and finite ($\leq k$) elsewhere. Accordingly, the Hough transform of k digital lines has a *maximum value* N at the k images of such lines, and value $< N$ elsewhere. It follows that the Hough transform reliably identifies multiple lines, even in the presence of noise, provided that $k \ll N$. As a point in case, we see that the Hough transform correctly identifies - as local maxima $H(s,i) = 4$ - the two lines $y = -x + 3/4$ and $y = 1 + x/3$, in our working example.

4: Digital Lines

Without loss of generality, assume that the *width* and *height* of our input domain is a power of two: $N = 2^n$. We restrict our attention to the $2N-1$ slopes of the form $s = h/(N-1)$, for h an integer in the range $-N < h < N$. The following table presents the four digital lines with $0 \leq h < 4$:



There are *many* ways to define the digital line $\mathcal{D}_n(s,i)$.

1. The iterative algorithm of Bresenham [1] defines $\mathcal{D}_n(s,i) = \mathcal{I}_n(h,0,i)$, where $\mathcal{I}_n$ can be expressed, for $h \geq 0$, by the formula:

$$\mathcal{I}_n(h,j,i) = \bigcup_{j \leq x < N} (x, i + \lfloor \frac{hx}{N-1} + \frac{1}{2} \rfloor).$$

In fact, some reputed textbooks use a *dual* definition, with rounding of the half-grid points *down* - as opposed to *up*:

$$\mathcal{I}_n(h,j,i) = \bigcup_{j \le x < N} (x, i + \lceil \frac{hx}{N-1} - \frac{1}{2} \rceil).$$

2. An alternative method $\mathcal{D}_n(s,i) = \mathcal{R}_n(h,0,i)$ recursively draws the two parallel half-segments:
   - from $(0,i)$ to $(\frac{N}{2}-1, i + \lfloor \frac{h}{2} \rfloor)$;
   - from $(\frac{N}{2}, i+h-\lfloor \frac{h}{2} \rfloor)$ to $(N-1, i+h)$.

   The corresponding algorithm is defined by the rules, where $N' = \frac{N}{2}$:

$$\begin{array}{rcl} \mathcal{R}_0(h,j,i) & = & (j,i) \\ \mathcal{R}_{n+1}(2h,j,i) & = & \mathcal{R}_n(h,j,i) \cup \mathcal{R}_n(h,j+N',i+h) \\ h>0:\ \mathcal{R}_{n+1}(2h+1,j,i) & = & \mathcal{R}_n(h,j,i) \cup \mathcal{R}_n(h,j+N',i+h+1) \\ h<0:\ \mathcal{R}_{n+1}(2h-1,j,i) & = & \mathcal{R}_n(h,j,i) \cup \mathcal{R}_n(h,j+N',i+h-1) \end{array}$$

A simple computer search shows that both methods coincide for $N \le 8$, and they differ at $\mathcal{I}_4(3/15,0)$, as shown by the table:

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\mathcal{I}_4(5,0,0)$	0	0	0	**1**	1	1	1	1	2	2	2	2	**2**	3	3	3
$\mathcal{R}_4(5,0,0)$	0	0	0	**0**	1	1	1	1	2	2	2	2	**3**	3	3	3

It is evident that both methods rarely differ; and, when they do, it is by *at most* one pixel. For the practical purpose of identifying straight lines in digital images, definitions $\mathcal{I}$ and $\mathcal{R}$ are equivalent: this is confirmed by computer simulation.

5: Fast Hough Transform

The recursive definition of $\mathcal{R}$ implies the following formula for computing $\mathcal{H}$:

$$\begin{array}{rcl} \mathcal{H}_0(h,j,i) & = & P(j,i) \\ \mathcal{H}_{n+1}(2h,j,i) & = & \mathcal{H}_n(h,j,i) + \mathcal{H}_n(h,j+N',i+h) \\ h>0:\ \mathcal{H}_{n+1}(2h+1,j,i) & = & \mathcal{H}_n(h,j,i) + \mathcal{H}_n(h,j+N',i+h+1) \\ h<0:\ \mathcal{H}_{n+1}(2h-1,j,i) & = & \mathcal{H}_n(h,j,i) + \mathcal{H}_n(h,j+N',i+h-1) \end{array}$$

We can read this as a *recursive* definition for $\mathcal{H}$. It can be implemented as such. Or, it can be obtained iteratively: starting from the input image $H_0 = \mathcal{P}$, compute H_1, where $h \in \{-1,0,1\}$; then, compute H_2, where $h \in \{-3,-2,-1,0,1,2,3\}$, and so on. In the general case, iterative step p computes the discrete Hough transforms of 2^{n-p} subdomains. Each subdomain has width 2^p, and we compute histograms for the $2^{p+1}-1$ values of h such that $|h| < 2^p$. The three stages in our working example, are:

$$H_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H_1 = \begin{bmatrix} 2 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 2 & 1 \\ 1 & 2 & 2 & 2 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

$$H_2 = \begin{bmatrix} 4 & 3 & 3 & 1 & 1 & 1 & 1 \\ 1 & 3 & 2 & 3 & 1 & 0 & 0 \\ 1 & 1 & 3 & 3 & 4 & 3 & 2 \\ 0 & 0 & 0 & 1 & 1 & 3 & 2 \end{bmatrix}$$

Proposition 1 *Computing all values $\mathcal{H}_n(h,0,i)$ for $0 \le i < N$, $N = 2^n$, and $|h| < N$ by the fast Hough transform FHT requires $2nN^2 - N(N-1)$ additions over $n+m$ bits numbers.*

For a square domain $N = 32$, the FHT requires 9K additions, compared to 64K additions for the naive Hough transform.

6: Circuits for the FHT

The smallest circuit for implementing the FHT uses *bit-serial* integer arithmetics; we denote by FHT_n the $N = 2^n$ version of the circuit.

The *input* to FHT_n is formed by the bits $P[0..N-1]$, which present in parallel the N points of a line in the input image. As we use bit-serial arithmetics, consecutive input lines are only consumed once, for each $n+m$ clock cycles.

The *output* from FHT_n is formed by $2N-1$ bits $H[-N+1..N-1]$. Line H bit-serially present - over $n+m$ consecutive cycles - the values of the Hough transform for all slopes in $-N < h < N$. Output lines which are $n+m$ time cycles apart correspond to consecutive values of the intercept parameter $0 \le i < N$.

Circuits for the *FHT* are built from only two primitive blocks:

1. Serial adder, with two inputs, one output and one internal storage bit for the carry - represented in our schemas by a circle around a + sign. The serial adder is a *primitive* operator in the ${}_2\mathbf{Z}$ language, noted by +.
2. Delay unit, implemented by a $n+m$ bit long shift register - represented by a square in our schemas. The corresponding ${}_2\mathbf{Z}$ definition of D is - with *reg* noting a one bit synchronous flip-flop:

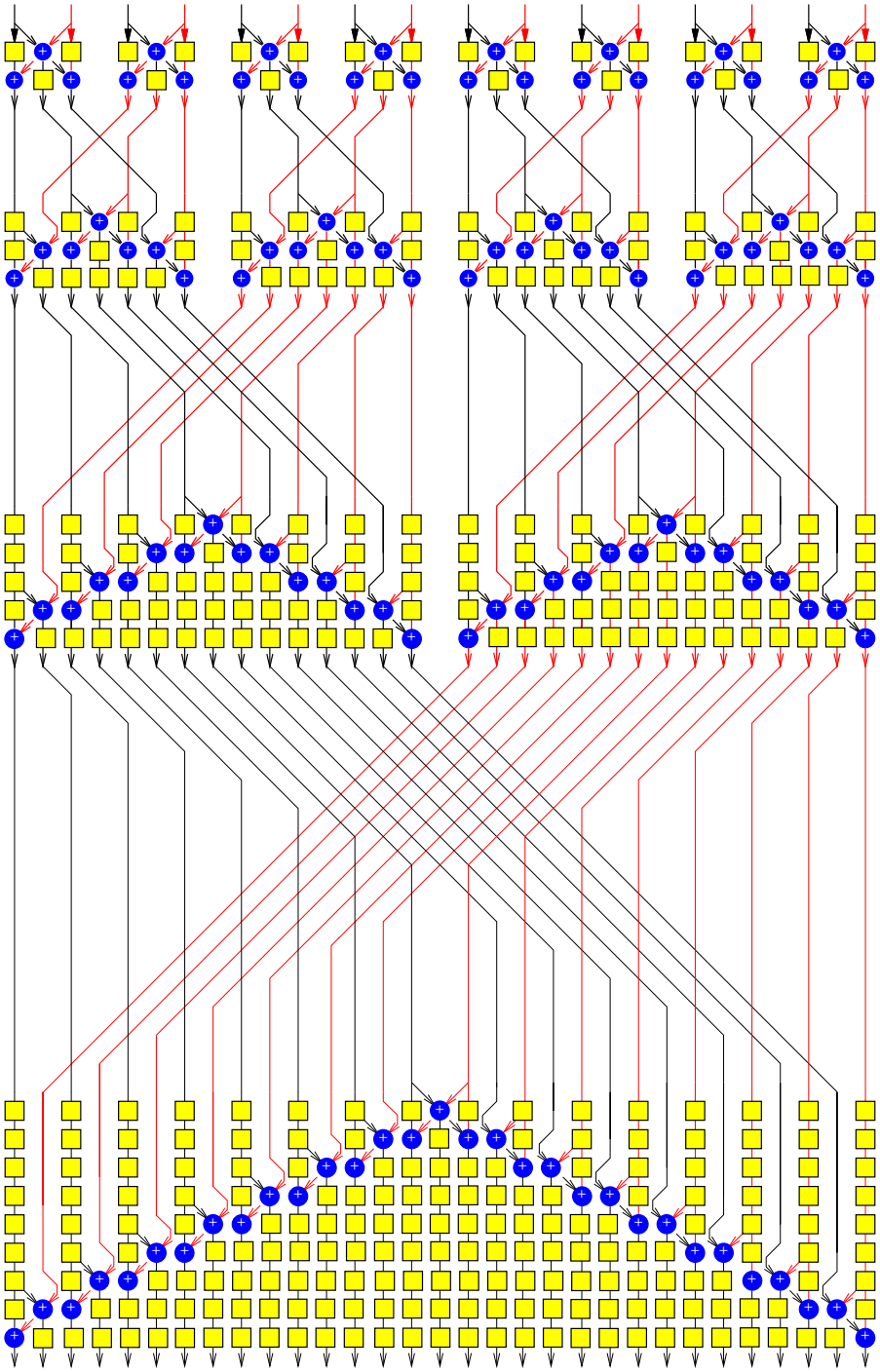
```
// m bits long shift register
D(m)(i) = o[m]
where
  o[0] = i;
  for k < m do
    o[k+1] = reg o[k]
  end for;
end where;
```

The basic operator in the FHT construction combines both circuits:

```
// Basic FHT butterfly operator
bFly(d,m, k)(a,b) = (l,r)
where
  da = D(d * m)(a);  // delay d
  l' = D(m)(da) + b;
```

```
  r' = D(m)(da + b);
  l = D(m * k)(l');   // delay k
  r = D(m * k)(r');
end where;
```

6.1: Schematics for FHT_4



6.2: A generator for FHT_n

expressed in the $_2$**Z** language [3].

```
// Fast Hough Transform
FHT(n, m)(in : [2**n]) = out : [2**(n+1) - 1]
where
  if n=0 then
    out = in
  else
    N     = 2**(n-1);
    low   = FHT(n-1, m)(in[0..N-1]);
    high  = FHT(n-1, m)(in[N..2*N-1]);
    // Negative slopes
    for k < N-1 do
       (out[2*k],out[2*k+1]) =
      bFly(N-k-1, m, k)(low[k], high[k]);
    end for;
    // Positive slopes
    for k < N-1 do
       (out[4*N-2*k-2], out[4*N-2*k-3]) =
      bFly(N-k-1, m, k)(high[2*N-k-2], low[2*N-k-2]);
    end for;
    // Minus one slope
  out[2*N-2] = D(m*(N-1))(D(m*N)(low[N-1])+high[N-1]);
    // Zero slope
  out[2*N-1] = D(m*N)(low[N-1] + high[N-1]);
    // Plus one slope
  out[2*N] = D(m*(N-1))(low[N-1] + D(m)(high[N-1]));
  end if
end where;
```

7: Conclusion

While it remains a compute intensive problem, we believe that the Hough transform will become much more widely used, thanks to the fast FHT algorithm, and the increase in computing power, for modern micro-processors and specific circuits.

8: Acknowledgements

Thanks to R. Bock for bringing the problem to our attention. H. Touati has programmed the methods, and performed the validating statistical analysis.

References

[1] Bresenham, J. E., *Algorithm for Computer Control of a Digital Plotter*, in *IBM Syst. J.*, 4(1):25-30, 1965.

[2] Badier J., Bock R., Busson P., Centro S., Charlot C., Davis E.W., Denes E., Gheorghe A., Klefenz F., Krischer W., Legrand I., Lourens W., Malecki P., Männer R., Natkaniec Z., Ni P., Noffz K.-H., Odor G., Pascoli D., Zoz R., Sobala A., Taal A., Tchamov N., Thielmann A., Vermeulen J., and Vesztergombi G., *Evaluating Parallel Architectures for two Real-Time Applications with 100kHz Repetition Rate*, in *IEEE Transactions Nuclear Science*, 40:1:45-55, 1993.

[3] Bourdoncle F., Vuillemin J. and Berry G., *The 2Z reference manual*, PRL report 40, Digital Equipment Corporation, Laboratoire de Recherche de Paris, 85 avenue Victor Hugo, 92563 Rueil Malmaison Cedex, France, 1994.

[4] Cormak A.M., *Representation of a function by its line integrals with some radiological applications*, in *J. Appl. Phys.* 34:2722-2727, 1963.

[5] Duda R.O. and Hart P.E., *Pattern Classification and Scene Analysis*, Wiley, New York, 1973.

[6] Jain Avril K., *Fundamentals of Digital Image Processing*, Prentice Hall, Englewood Cliffs, 1989.

[7] Radon J., *Uber die Bestimmung von Funktionen durch ihre Integralwerte langs gewisser Mannigfaltigkeiten*, Ber. Saechsische Akad. Wiss. 29:262-279, 1917.

[8] Rosenfeld A., Kak A.C., *Digital Picture Processing*, Academic Press, San Diego, 1982.