Considerations for hardware Hough transforms

Donald G. Bailey
School of Engineering and Advanced Technology
Massey University
Palmerston North, New Zealand
D.G.Bailey@massey.ac.nz

***Abstract*—Three line parameterisations for the Hough transform are compared. The angle and line resolution of each parameterisation is derived for a given number of parameter bins. The computational implications of this are discussed, with proposal for a parameterisation based on parallel coordinates outlined.**

Keywords-Hough transform; parallel coordinates; resolution; FPGA

I. Introduction

The Hough transform is a well known technique for detecting lines (and other parameterised shapes) within an image. The key concept of the Hough transform is to parameterise the shape being detected, and to transform from image space to parameter space. For detecting lines, each line in image space is represented by a unique point in parameter space corresponding to the parameters of the line. The image is preprocessed to detect edge pixels. Then each detected edge pixel votes for candidate parameters corresponding to every possible line that passes through that point. Multiple collinear points will accumulate in the parameter space bin associated with the line through the points. Parameter space is therefore examined for peaks corresponding to clusters of edge pixels which share common parameters.

While much has been written of the Hough transform and its many variations (see for example [1, 2]), this paper will focus on some of the requirements pertinent to an FPGA based implementation of the Hough transform for detecting lines. Section II reviews the three main line parameterisations. The key challenges to designing an efficient FPGA based implementation are elaborated in Section III. These are to minimise the memory required to achieve the necessary resolution, and to minimise the computational complexity of the algorithm. Therefore, Section IV analyses the resolution which may be achieved for a given size of sample space for each of the parameterisations. Section V discusses the computational implications and approaches which may be used by each of the methods.

The novel contribution of this paper is the resolution analysis of the different parameterisations, and the implications of this for hardware implementation.

II. Line Parameterisations

Image coordinates are used as shown in Figure 1. The origin is in the top left corner of the image which has $H \times V$ pixels. In some situations, it is advantageous to offset the origin to the centre of the image:

$$(x_c, y_c) = (x, y) - \tfrac{1}{2}(H, V) \tag{1}$$

The angle of a line, φ, is defined as clockwise from horizontal, with a sample point on the line, (x_i, y_i).

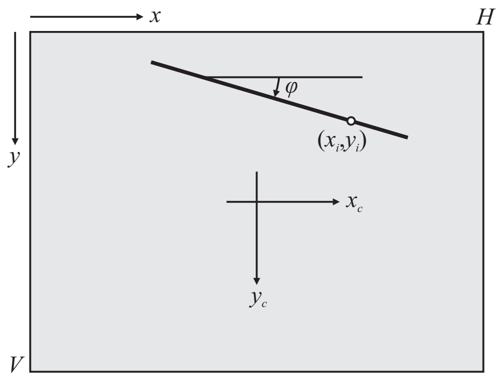

Figure 1. Coordinate system and basic definitions.

A. Hough's Original Transform

Hough's original transform [3] was based on a slope-intercept parameterisation:

$$y = mx + c \tag{2}$$

Each detected pixel (x_i, y_i) within the image votes for points along a linear trace in $\{m, c\}$ parameter space

$$c = -mx_i + y_i \tag{3}$$

This parameterisation works well for lines which are approximately horizontal. However, as the line becomes vertical, both the slope and the intercept tend towards ±infinity. In fact it can be shown that the parameter space will always be unbounded [4] for any linear parameterization covering lines of all possible orientations.

This problem may easily be overcome by having two sets of parameter spaces [3], one for horizontal lines, and one for vertical lines:

$$\begin{aligned} c &= -m_1 x_i + y_i, \quad -1 \le m_1 < 1 \\ c &= -m_2 y_i + x_i, \quad -1 \le m_2 < 1 \end{aligned} \tag{4}$$

B. *Standard Hough Transform*

An alternative approach is to change the parameterisation to the length and angle of the normal to the line (see Figure 2.), as first proposed by Duda and Hart [5]:

$$\rho = x\cos\theta + y\sin\theta \tag{5}$$

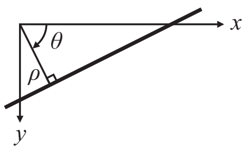

Figure 2. The standard $\{\rho,\theta\}$ parameterisation.

Rather than each edge point voting along a line in parameter space, each point votes along a sinusoidal trace. The periodic nature of sinusoids means that parameter space can be restricted to $0° \le \theta < 180°$ (or $-90° \le \varphi \le 90°$).

C. *Parallel Coordinates*

An alternative parameterisation that has been proposed recently is based on parallel coordinates [6, 7]. Each detected point in the image votes for parameters along the line between the x component on one side of parameter space, and the y component on the other side (hence the name parallel coordinates), as shown in Figure 3. A set of collinear points in image space will result in a set of intersecting traces within parameter space.

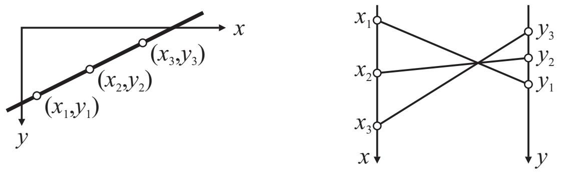

Figure 3. Parallel coordinate representation of a line

Traces for points along a line intersect at a common point because of proportionality. Any pair of points on the line will have the same ratio of $\Delta x : \Delta y$. Therefore in parallel coordinate space, this will result in a pair of triangles with bases Δx and Δy of the same proportion.

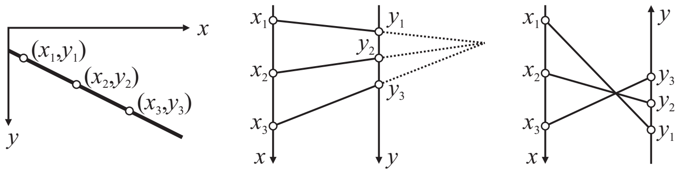

Figure 4. Failure of parallel coordinates and solution by flipping one axis.

As with the original Hough transform, detected image points vote along a line within parameter space. Therefore, to be able to represent all possible lines, the parameter space will be unbounded [4]. The parameterisation described above will only intersect between the parallel coordinates for $-90° \le \varphi \le 0°$. With $0° < \varphi < 90°$, the intersection will be outside the coordinates, as shown in Figure 4. , and potentially up to an infinite distance away $(\varphi = 45°)$. This problem can be overcome by introducing a second set of parallel coordinates with a flipped x or y axis [7].

Parallel coordinates can readily be converted to the more usual two orthogonal axes. Let v be the distance along the parallel coordinates, and α be the proportion between the x and y parallel coordinates. An edge point may be mapped along the lines in $\{\alpha, v\}$ parameter space

$$\begin{aligned} v &= (1-\alpha_1)x_i + \alpha_1 y_i \quad 0 \le \alpha_1 \le 1 \\ v &= (1-\alpha_2)x_i + \alpha_2 \bar{y}_i \quad 0 \le \alpha_2 \le 1 \end{aligned} \tag{6}$$

where $\bar{y}_i$ is the flipped coordinate. With the origin in the top-left corner:

$$\bar{y}_i = V - y_i \tag{7}$$

Or with the origin in the centre of the image:

$$\bar{y}_i = -y_i \tag{8}$$

III. Hardware Implementation Considerations

There are two key challenges in designing an efficient implementation. The first is a memory bandwidth bottleneck. Typically, 10% of the pixels in the input image are edge pixels [8], and each detected pixel casts many votes in parameter space. Each vote requires two memory accesses, first to read the count within a bin, and second to write the incremented count.

When processing with an FPGA, the input image is usually streamed, which places a tight timing constraint on the processing. This makes it difficult, if not impossible, to perform the Hough transform in real time with a single off-chip memory.

The bandwidth bottleneck must be overcome through using distributed memory on the FPGA. The many small on-chip memory blocks can be accessed independently in parallel, greatly increasing the available bandwidth. However, on-chip memory is a relatively scarce resource, therefore one motivation is to minimise the memory required to achieve the necessary resolution in parameter space.

On-chip memory introduces a further constraint. The memory blocks are generally a power of two in depth, and a power of two multiple of 9 bits in width. To make optimal use of the memory, the constraints this places on parameter space are as follows:

- Each accumulator bin is either 9 bits (counts 0-511) or 18 bits (counts 0-262143) in width.
- The number of bins in the c, ρ, or v ‘length’ axis will be a power of 2. For 9kbit blocks, this will be 1024 or 512.
- The number of bins in the m, θ, or α ‘angle’ axis is arbitrary, and limited by the number of available memory blocks.
- Dual-port memory enables the read-increment-write in a single cycle using separate ports for read and write.

For small to medium size images, the main constraint is the number of bins in the angle axis. For high resolution images, both axes are constrained, and it may be necessary to perform multiple passes, calculating a region of parameter space with each pass (see for example [9, 10]). This requires additional memory for buffering the detected pixels for later passes, although this can be off-chip.

The alternative to multiple passes is to quantise parameter space more coarsely. The quantisation error is related to the length of the line segment [11], so knowledge of the problem is also required to ensure that the quantisation is adequate. An important factor is to determine the effect of quantisation on angle and line resolution, so that the memory requirements may be reduced without sacrificing the resolution.

A second consideration is the computational complexity of the Hough transform. In hardware, each calculation must be implemented in hardware. Therefore reducing the computational complexity is required to achieve an efficient implementation. Many architectures have been proposed for reducing the computational burden on FPGAs, including CORDIC arithmetic to calculate the trigonometric functions [12], using a logarithmic number system to convert multiplications to additions [13]. Another common approach is to use incremental arithmetic to replace the trigonometric functions and multiplications by incremental additions [14, 15].

IV. Resolution Analysis

One of the keys of implementing a distributed memory based parameter space accumulator on an FPGA is to reduce the memory requirements to obtain a given level of accuracy. The purpose of this section is to analyse the angle and line resolutions for a given number of bins.

A. *Duda and Hart's Hough Transform*

Since the θ axis directly represents the line angle, with uniform quantisation, the Hough transform has uniform angular resolution. Let there be N_θ angle bins, over the range 0 to 180 degrees, then the angle resolution will be:

$$\Delta\varphi_{DH} = \frac{180°}{N_\theta} \tag{9}$$

The line resolution is measured by the perpendicular spacing of the lines corresponding to each bin along the length axis. Since ρ is perpendicular to the line, and spans from $-\rho_{\max}$ to $\rho_{\max}$ (the maximum normal length) the bins are spaced with

$$\Delta l_{DH} = \frac{2\rho_{\max}}{N_\rho} \tag{10}$$

This can be minimised by offsetting the origin to the centre of the image, giving

$$\Delta l_{DH} = \frac{\sqrt{H^2+V^2}}{N_\rho} \tag{11}$$

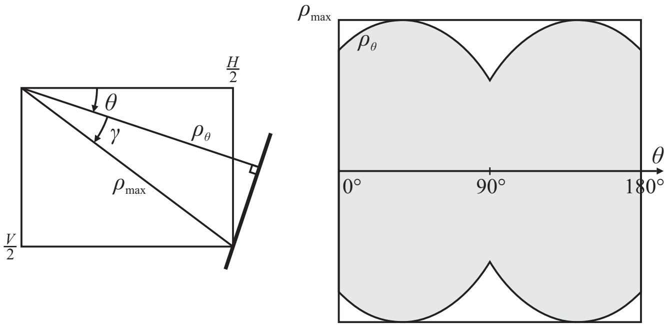

Figure 5. Envelope within Hough parameter space.

The maximum radius is only reached at certain angles, as demonstrated in Figure 5. The maximum radius as a function of angle for $0° \le \theta \le 90°$ is given as

$$\begin{aligned}\rho_\theta &= \rho_{\max}\cos\gamma \\ &= \rho_{\max}\cos\left(\tan^{-1}\frac{V}{H}-\theta\right) \\ &= \frac{H}{2}\cos\theta + \frac{V}{2}\sin\theta\end{aligned} \tag{12}$$

and similarly for the other quadrants. This allows the line resolution to be increased, by stretching the ρ axis by $\rho_{\max}/\rho_\theta$ [16]. The line resolution of the stretched parameter space is therefore

$$\Delta l_{DHs} = \frac{H\left|\sin\varphi\right| + V\left|\cos\varphi\right|}{N_\rho} \tag{13}$$

B. *Original Hough Transform*

With Hough's original transform, it would be more usual to have equal quantisation steps along the m axis. While the average angle resolution would be the same as for the Duda and Hart transform, it is no longer constant. From Figure 1. it is evident that

$$m_1 = \tan\varphi \tag{14}$$

Since the parameter space is split into two regions, half of the angle samples will correspond to each region. Therefore

$$\begin{aligned}\Delta\varphi_H &= \Delta m_1 \frac{d\varphi}{dm_1} = \Delta m_1 \frac{d\tan^{-1} m_1}{dm_1} \\ &= \frac{2}{N_m/2}\frac{180}{\pi}\frac{1}{1+m_1^2} = \frac{720\cos^2\varphi}{\pi N_m}\end{aligned} \tag{15}$$

and similarly for the other quadrant. Therefore best angle resolution is obtained for diagonal lines, at the expense of poorer resolution for horizontal and vertical lines.

With the origin in the top left of the image, the intercept axis spans from $-H$ to $H+V$, although there is a lot of unused space, as shown in Figure 6. Moving the origin to the centre of the image significantly reduces the waste space.

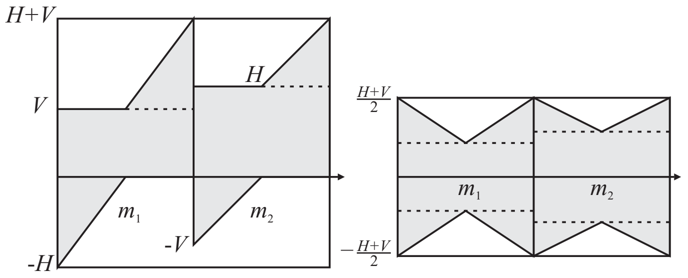

Figure 6. Parameter space extent. Left: origin in top left; Right: origin in centre of image.

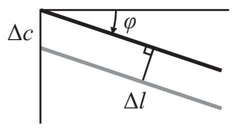

Figure 7. Line resolution dependence on angle.

Since the lines are not perpendicular to the intercept, quantisation of the intercept axis will result in an angle dependence of the line resolution. This is illustrated in Figure 7. where the line resolution improves as the line angle approaches 45 degrees:

$$\Delta l_H = \Delta c \cos\varphi = \frac{H+V}{N_c}\cos\varphi \tag{16}$$

for $\{m_1, c\}$ and similarly for $\{m_2, c\}$:

$$\Delta l_H = \frac{H+V}{N_c}\sin\varphi \tag{17}$$

As with the Duda and Hart transform, the line resolution can be improved at some angles stretching the length axis to fill in the unused space. Unfortunately, simply scaling the c axis means that the transform is no longer linear:

$$c = \frac{(y - m_1 x)(V+H)}{V + |m_1| H} \tag{18}$$

and therefore loses its computational simplicity.

C. Parallel Coordinate Hough Transform

With parallel coordinates, the α axis represents the line angle with

$$\tan\varphi = \frac{\alpha_1}{1-\alpha_1} \tag{19}$$

As with the linear Hough transform, the angle resolution is not constant for a constant step-size in α. The angle resolution is therefore

$$\begin{aligned}\Delta\varphi_{PC} &= \Delta\alpha_1 \frac{d\varphi}{d\alpha_1} = \Delta\alpha_1 \frac{d}{d\alpha_1}\tan^{-1}\frac{\alpha_1}{1-\alpha_1} \\ &= \frac{1}{N_\alpha/2}\frac{180}{\pi}\frac{1}{(1-\alpha_1)^2+\alpha_1^2} = \frac{360(1+\sin 2\varphi)}{\pi N_\alpha}\end{aligned} \tag{20}$$

and similarly for α_2. Best angle resolution is obtained for horizontal and vertical lines at the expense of poorer angle resolution along the diagonals.

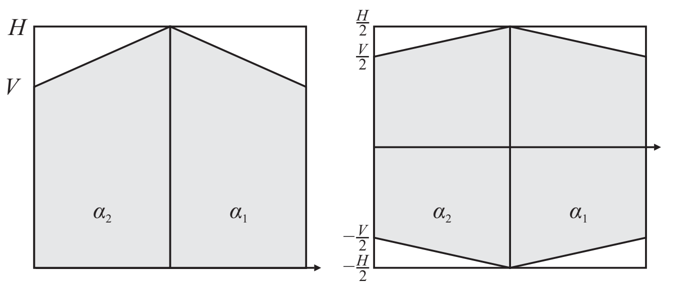

Figure 8. Parameter space extent with parallel coordinates. Left origin in top left; Right: origin in centre of image.

Determining the line resolution is a little more complex. First, the extent of the parameter space is independent of whether the origin is in the corner or centre of the image, as demonstrated in Figure 8. Of all the parameter spaces, it is the most compact, extending only the maximum of H or V.

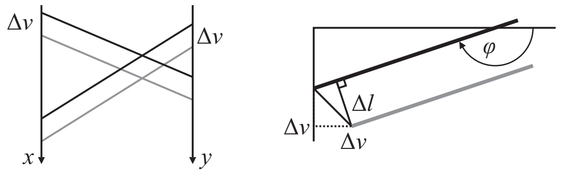

Figure 9. Line resolution dependence on angle.

As shown in Figure 9. , an offset of Δv in parameter space is equivalent to an offset along both the x and y directions. Therefore, in the $\{\alpha_1, v\}$ space

$$\begin{aligned}\Delta l_{PC} &= \sqrt{2}\Delta v \cos(\varphi - 135) \\ &= \frac{H}{N_v}(\sin\varphi - \cos\varphi)\end{aligned} \tag{21}$$

and similarly for $\{\alpha_2, v\}$

$$\Delta l_{PC} = \frac{H}{N_v}(\sin\varphi + \cos\varphi) \tag{22}$$

The unused parameter space may be used to improve the line resolution in two ways. The first, by stretching the v axis, suffers the same problem as with the stretched Hough transform, in that it is no longer linear. However, with parallel coordinates, stretching the y axis to make the image square before applying the parallel coordinate transform will avoid unused space and retain a linear transform. This will also affect the angle resolution, since (19) becomes

$$\tan\varphi = \frac{\frac{H}{V}\alpha_{1s}}{1-\alpha_{1s}} \tag{23}$$

Recalculating the angle resolution gives

$$\Delta\varphi_{PCs} = \frac{360\left(H\left|\cos\varphi\right| + V\left|\sin\varphi\right|\right)^2}{\pi H V N_\alpha} \tag{24}$$

with the corresponding line resolution given by

$$\Delta l_{PCs} = \frac{H\left|\sin\varphi\right| + V\left|\cos\varphi\right|}{N_v} \tag{25}$$

It is interesting to note that the line resolution is the same as for the stretched Duda and Hart transform (equation (13)).

The second approach to avoiding waste space with the parallel lines transform is to pack the two subspaces into a single rectangular region. This is easiest with the origin in the top-left corner of the image. Packing has no effect on angle resolution, however it allows a finer quantisation of the v axis giving

$$\Delta l_{PCp} = \frac{H+V}{2N_v}\left(\left|\sin\varphi\right| + \left|\cos\varphi\right|\right) \tag{26}$$

D. Summary

The angle and line resolutions of the different schemes are compared in Figure 10. and Figure 11. respectively.

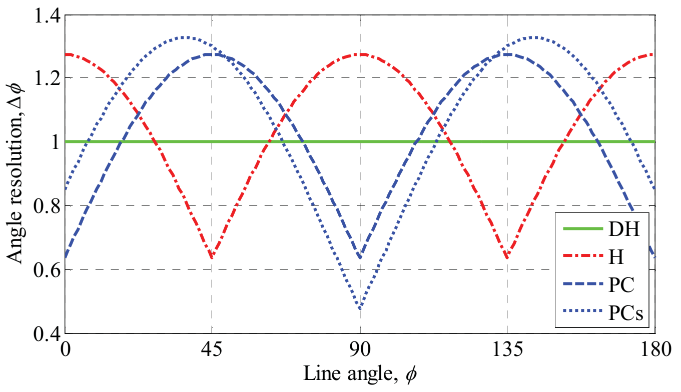

Figure 10. Angle resolution as a funtion of angle for 4:3 aspect ratio, normalised to 1 degree per sample $(N_\theta = N_m = N_\alpha = 180)$.

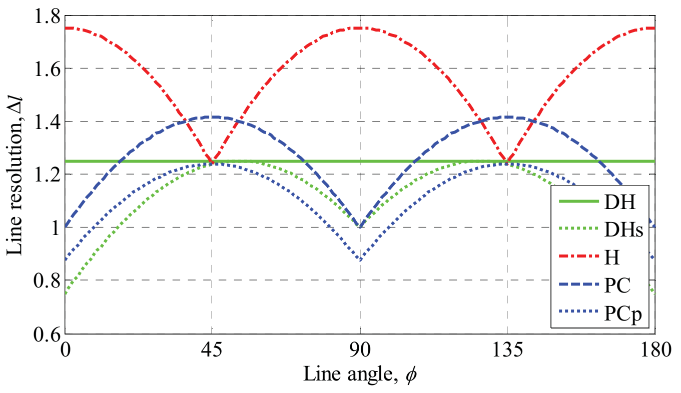

Figure 11. Line resolution as a function of angle for 4:3 aspect ratio, notmalised to 1 pixel per sample $(N_\rho = N_c = N_v = H)$.

While the average angle resolution is the same for all methods, only the Duda and Hart transform has uniform resolution with uniform sampling along the 'angle' axis. The increased angle resolution at certain angles could be of advantage if those angles were of particular interest. Otherwise variable angle resolution is a limitation of the linear methods.

To give a more uniform angle resolution, the m or α axes could be sampled non-uniformly, for example to give the angle intercept ($\{\varphi, c\}$) space which is sometimes used [8]. The disadvantage is that such schemes lose the computational simplicity which results from linear traces.

The line resolution as a function of angle is particularly interesting. The results shown in Figure 11. agree with the experimental results obtained by Dubska et al [7], especially since they only considered the case of square images $(H = V)$, which gives the curve identical to Δl_{PCp}. The compactness of the parallel coordinates parameter space gives it a distinct advantage even over the conventional Hough transform. In addition, it is a linear parameter space which results in a computational simplicity. For images which are not square, optimal resolution can be obtained by either stretching or packing. In either case, linearity is still maintained.

V. Computational Considerations

With the conventional Duda and Hart transform, the computationally expensive step is calculating the sine and cosine of the angle. Incremental calculation of the sine and cosine (for example using the CORDIC principle) is not generally practical because the accumulated error with each iteration soon becomes excessive. Since only a small number of predetermined angles are used, the sine and cosine can be calculated in advance and stored in lookup tables. Although direct evaluation of (5) requires two multiplications for each angle, they can be reused for a three additional angles because of symmetry (a total of $N_\theta / 2$ multiplications). If the stretched parameter space is used, the scaling factor can simply be incorporated into the values within the lookup tables (although then the scaled sine and cosine can only be used for one additional angle – N_θ multiplications).

When distributed memory is used for parameter space, the use of lookup tables for calculating sine and cosine creates an additional bandwidth bottleneck. Rather than using a table, the sine and cosine values are distributed, requiring separate multipliers, two for every four angles as described above. On FPGAs where hardware multipliers are plentiful, this is not too much of an issue, although on devices without many multipliers, this can be expensive in terms of resources.

With the linear Hough transform, direct evaluation of (4) requires one multiplier for every two slopes (one positive and one negative – a total of $N_m / 2$ multiplications). Since the trace in parameter space of each detected edge point is a straight line, incremental calculation can be used:

$$x_i(m + \Delta m) = x_i m + x_i \Delta m \tag{27}$$

with $x_i \Delta m$ calculated once for each trace. It is even simpler if N_m is a power of 2, since then multiplication by Δm is simply a bit shift. Incremental calculation can still be used with distributed memory, simply by pipelining each stage, taking a total of $N_m / 4$ clock cycles to increment all accumulators.

A similar argument can be applied to the parallel coordinates scheme. Equation (6) can be rearranged to require only one multiplication for every two angles

$$v = \frac{x_i + y_i}{2} \pm (y_i - x_i)(\alpha - \tfrac{1}{2}) \tag{28}$$

When using distributed memory, an alternative incremental calculation scheme is also possible. With a streamed input, x_i increments only with every row, and along a row y_i is incremented. A double incremental calculation can be used to for a given memory block (associated with a value of α) to derive successive addresses, v_α . Moving between rows

$$v_\alpha(x_i + 1, 0) = v_\alpha(x_i, 0) + \frac{1-\alpha}{\Delta v} \tag{29}$$

and from one pixel to the next in a row

$$v_\alpha(x_i, y_i + 1) = v_\alpha(x_i, y_i) + \frac{\alpha}{\Delta v} \tag{30}$$

Since several successive pixels will access the same memory location, especially near the x axis, a simple caching arrangement may be used to halve the memory bandwidth required. This makes the packed memory arrangement practical.

A further optimisation is to make use of the edge orientation. Each detected edge pixel usually has an associated gradient direction which can be used to limit the number of 'angles' for which the detected pixel votes [12, 17, 18]. This serves two purposes. First, it reduces the memory bandwidth required, making a global memory implementation possible. Second, it reduces unnecessary clutter within the parameter space, making peak detection more reliable.

VI. Conclusions

This paper has reviewed three different parameterisations for detecting lines using the Hough transform. While the commonly used normal length and angle parameterisation introduced by Duda and Hart has a certain mathematical elegance, the traces in parameter space are nonlinear, making incremental calculation more difficult. The linear Hough transform and parallel coordinate transforms computationally simpler, although they require multiple parameter spaces. With distributed memory, however, this is not an issue.

The main disadvantage of the linear parameterisations is that the angle resolution is not uniform. The linear Hough transform has better resolution along diagonal lines, at the expense of horizontal and vertical lines. The opposite is true of the parallel coordinate transform. Of the three parameterisations, it is shown that the linear Hough transform has the poorest line resolution for a given number of accumulator bins, while the parallel coordinate transform gives the best performance. This is because the parallel coordinate representation is the most compact, and has the least unused accumulator space, especially for square images. With rectangular images, the unused space can be exploited either by stretching or packing, while still using linear traces.

Finally, a new double increment scheme is proposed for use with parallel coordinates which both reduces the computation and also memory bandwidth required. Therefore, it is concluded that the parallel coordinate parameterisation would give the best overall performance.

References

[1] V. F. Leavers, "Which Hough transform?," *Computer Vision, Graphics, and Image Processing: Image Understanding,* vol. 58, no. 2, pp. 250-264, 1993.

[2] J. Illingworth and J. Kittler, "A survey of the Hough transform," *Computer Vision, Graphics, and Image Processing,* vol. 44, no. 1, pp. 87-116, 1988.

[3] P. V. C. Hough, "Method and means for recognizing complex patterns," *United States of America* patent 3069654, 1962.

[4] P. Bhattacharya, A. Rosenfeld, and I. Weiss, "Point-to-line mappings as Hough transforms," *Pattern Recognition Letters,* vol. 23, no. 14, pp. 1705-1710, 2002.

[5] R. O. Duda and P. E. Hart, "Use of the Hough transformation to detect lines and curves in pictures," *Communications of the ACM,* vol. 15, no. 1, pp. 11-15, 1972.

[6] A. Inselberg, A. Chatterjee, and B. Dimsdale, "System using parallel coordinates for automated line detection in noisy images," *United States of America* patent 5631982, 1997.

[7] M. Dubska, A. Herout, and J. Havel, "PClines - line detection using parallel coordinates," in *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Colorado Springs, Colorado, USA, 2011, pp. 1489-1494.

[8] M. G. Albanesi, M. Ferretti, and D. Rizzo, "Benchmarking Hough transform architectures for real-time," *Real-Time Imaging,* vol. 6, no. 2, pp. 155-172, 2000.

[9] N. Nagata and T. Maruyama, "Real-time detection of line segments using the line Hough transform," in *IEEE International Conference on Field-Programmable Technology*, Brisbane, Australia, 2004, pp. 89-96.

[10] R. Jošth, M. Dubská, A. Herout, and J. Havel, "Real-time line detection using accelerated high-resolution Hough transform," in *17th Scandinavian Conference on Image Analysis*, Ystad, Sweden, 2011, pp. 784-793.

[11] T. M. van Veen and F. C. A. Groen, "Discretization errors in the Hough transform," *Pattern Recognition,* vol. 14, no. 1-6, pp. 137-145, 1981.

[12] S. M. Karabernou, L. Kessal, and F. Terranti, "Real-time FPGA implementation of Hough transform using gradient and CORDIC algorithm," *Image and Vision Computing,* vol. 23, no. 11, pp. 1009-1017, 2005.

[13] P. Lee and A. Evagelos, "An implementation of a multiplierless Hough transform on an FPGA platform using hybrid-log arithmetic," in *Real-Time Image Processing 2008*, San Jose, California, USA, 2008, pp. 68110G-1-10.

[14] H. Koshimizu and M. Numada, "FIHT2 algorithm: A fast incremental Hough transform," in *IAPR Workshop on Machine Vision Applications*, Tokyo, Japan, 1990, pp. 233-236.

[15] S. Tagzout, K. Achour, and O. Djekoune, "Hough transform algorithm for FPGA implementation," *Signal Processing,* vol. 81, no. 6, pp. 1295-1301, 2001.

[16] L. da Fontoura Costa and M. B. Sandler, "Improving parameter space for Hough transform," *Electronics Letters,* vol. 25, no. 2, pp. 134-136, 1989.

[17] F. O'Gorman and M. B. Clowes, "Finding picture edges through collinearity of feature points," *IEEE Transactions on Computers,* vol. 25, no. 4, pp. 449-456, 1976.

[18] R. Cucchiara, G. Neri, and M. Piccardi, "A real-time hardware implementation of the Hough transform," *Journal of Systems Architecture,* vol. 45, no. 1, pp. 31-45, 1998.