

Design of a Real-time Hough Transform on Configurable Logic

Emeric K. Jolly* and Martin Fleury
Dept. of Electronic Systems Engineering, University of Essex
{ekjol,fleum}@essex.ac.uk

1 Introduction

The Hough transform (HT) is frequently used to locate possibly occluded straight edges or lines in machine vision. Each detected edge pixel in a binary image votes for a potential edge upon which it might lie. The HT is potentially suitable for video-rate applications such as motion detection (by comparison of successive HT transformed frames), but the computational burden is high, motivating hardware implementations. There have been a number of HT implementations using the CORDIC algorithm which allows trigonometric angles to be generated primarily by shifts and additions and which can be adapted to find HT radii. However, CORDIC generates a series of converging micro-rotations, whereas we have found that a series of angles can be swept out, thus further increasing the throughput, which is still slow in traditional CORDIC. By dividing up the angle range, our method can incrementally improve the angular resolution and accuracy, without the need for scaling factors and with uni-directional rotations.

The successful implementation was achieved on an XC2V3000 Virtex-2, which is one of the large capacity SRAM-based platform FPGAs from Xilinx. The Handel-C hardware compiler running within Celoxica's DK2/3 IDE was used to develop the design on an RC203 board. The RC203 board is one of Celoxica's range of development boards, while a low-cost version for educational usage, the RC10¹ has recently been released. (The alternative Digilent XUP V2P "Curriculum on a Chip" also comes into contention.²) In fact, early work on the real-time HT took place on a low-cost Digilent board, before transferring to an RC1000-PP (Virtex 1), and, thence, the Handel-C code was ported to the RC203 system.

Handel-C uses hardware templates to model program constructs and, hence, there has been a concern over resource usage. In fact, the compiled design used less than 7% of the Virtex-2 (1,113 slices) and ran at 50 MHz. If the angle range is divided into two, then the real-time HT accomplishes 26 frame/s operating on 256×256 pixel images at an angular resolution of 0.9° . Accuracy improves as the angle range is successively divided up, but even without division, the error is only a few per cent, sufficient for typical applications. Currently, on-chip block RAM is utilized to store the vote arrays (divided into sectors to allow concurrent access). However, as this resource is limited quantization of the votes will become necessary for scaled-up designs.

*This work was accomplished through a Nuffield Bursary, which are awarded to prepare students for research.

¹Refer to <http://www.celoxica.com/techlib/files/CEL-W050518YVZ-348.pdf>

²Refer to <http://www.digilentinc.com/info/XUPV2P.cfm>.

2 HT components

2.1 CORDIC algorithm

Consider coordinate origins at the geometrical center of an image, then seek the length of a radius R from the origin normal to a straight edge passing between a detected pixel with Cartesian coordinates (X, Y) . R subtends an angle θ so that

$$R = X \cos \theta + Y \sin \theta, \quad (1)$$

with θ valid from 0 to 2Π .

In practice, θ is ranged from 0 to Π as the sign of R distinguishes the position of R . Dividing through by $\cos \theta$ and with suitable trigonometric substitutions yields:

$$R_{1X} = (X + Y \tan \theta) \cos \theta, 0 < \theta < \frac{\Pi}{2}, \Pi < \theta < \frac{3\Pi}{2} \quad (2)$$

$$R_{1Y} = (Y - X \tan \theta) \cos \theta, \frac{\Pi}{2} < \theta + \frac{\Pi}{2} < \Pi, \frac{3\Pi}{2} < \theta + \frac{\Pi}{2} < 2\Pi, \quad (3)$$

which are also the basis of the CORDIC algorithm for hardware calculation of trigonometric values by means of micro-rotations. By varying θ , all possible values of R are found for varying straight lines through invariant (X, Y) . θ can be replaced by $\arctan \phi$, so that $\tan(\arctan \phi) = \phi$.

In the traditional CORDIC [1] applied in [4][2], $\phi = \pm 2^{-i}$ with $i = 0, 1, 2, \dots$, allowing a set of micro-rotations to converge to a given angle. However, this requires a look-up table to establish the direction of a micro-rotation at any stage in the iteration and a scaling re-adjustment [2].

2.2 Constant rotation CORDIC-like transform

For this application, it turns out that the micro-rotations can be uni-directional and of constant size causing a series of angles to be generated. If the step increment $\Delta\theta$ is 1.79° then $\phi = 2^{-5}$. The terms multiplied by 'tan θ ' will result in a simple right shift operation by five places. Then

$$\frac{R_{1X}}{2\Delta\theta} \approx \frac{R_{1X}}{\Delta\theta} + \frac{R_{1Y}}{\Delta\theta} \tan \Delta\theta, \frac{R_{1Y}}{2\Delta\theta} \approx \frac{R_{1Y}}{\Delta\theta} - \frac{R_{1X}}{\Delta\theta} \tan \Delta\theta, \quad (4)$$

as, for sufficiently small $\Delta\theta$, the gain, $\cos \Delta\theta$, approaches one. Fig. 1 shows a hardware cell to calculate successive values.

2.3 Multi-Sector transform

The Multi-Sector transform (MST) uses the same micro-rotations but several cells are employed to process the angle range and generate radii. A simple MST for instance, will have one cell of the type shown in Fig. 1 for θ incrementing and a second variant cell to calculate

$$\frac{R_{2X}}{2\Delta\theta} \approx \frac{R_{2X}}{\Delta\theta} - \frac{R_{2Y}}{\Delta\theta} \tan \Delta\theta, \frac{R_{2Y}}{2\Delta\theta} \approx \frac{R_{2Y}}{\Delta\theta} + \frac{R_{2X}}{\Delta\theta} \tan \Delta\theta, \quad (5)$$

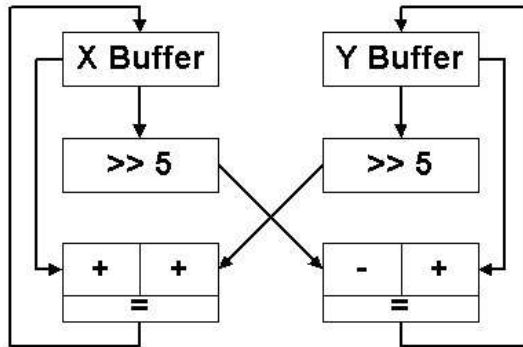


Figure 1: Simple CORDIC-like cell for calculating radii values in successive micro-rotation increments of 1.79° .

for θ decrementing, resulting in double the throughput.

The initial radii inputs to the two cells (refer to Fig. 2) are calculated from the original X and Y pixel coordinates (relative to the image center). For the simple MST, initialize these values with the radii corresponding to an offset of $\frac{\Pi}{4}$. This can be achieved with a series of shifts and additions to the original pixel coordinates. To maintain sufficient accuracy for machine vision applications the values stored in the R_{1X} , R_{1Y} , R_{2X} and R_{2Y} registers are $\frac{1}{\tan \Delta\theta} = \frac{1}{\phi}$ times larger than the actual values, for example $\frac{1}{\phi} = 2^5 = 32$. Thus, one must initialize $R_{1x} = R_{2x}$ from:

$$32 \times X \cos\left(\frac{\Pi}{4}\right) + 32 \times Y \sin\left(\frac{\Pi}{4}\right) = (X + Y) \times 32 \times \frac{\sqrt{2}}{2} \approx 22.6 \times (X + Y) \quad (6)$$

and equivalently for $R_{1y} = R_{2y}$. As $16 + 4 + 2 + 0.5 = 22.5$, which is close to 22.6, the shifts and adds applied to form (6) are $X \lll 4 + X \lll 2 + X \lll 1 + X \ggg 1$.

2.4 MST pipeline

A block diagram of an MST is given in Fig. 3. It is possible to form a three stage pipeline consisting of: (1) Calculate radii; (2) Read and increment vote; (3) Write vote back. Therefore, a further improvement in throughput arises from a pipelined MST (PMST) if on-chip, dual-port block RAMs, allowing simultaneous read and writes, are employed to store votes. On-chip block RAMs have been shown [3] to be an effective addition to FPGA architecture.

3 Results

The algorithm was performed on a Virtex-II XC2V3000 FPGA. A standard image 256×256 -pixel image, "airplane"³, Fig. 4, with clear linear features, the runway borders, was employed. An edge image was then binary thresholded, Fig. 5, to produce

³Available from the USC-SIPI Image Database at <http://sipi.usc.edu/database/>.

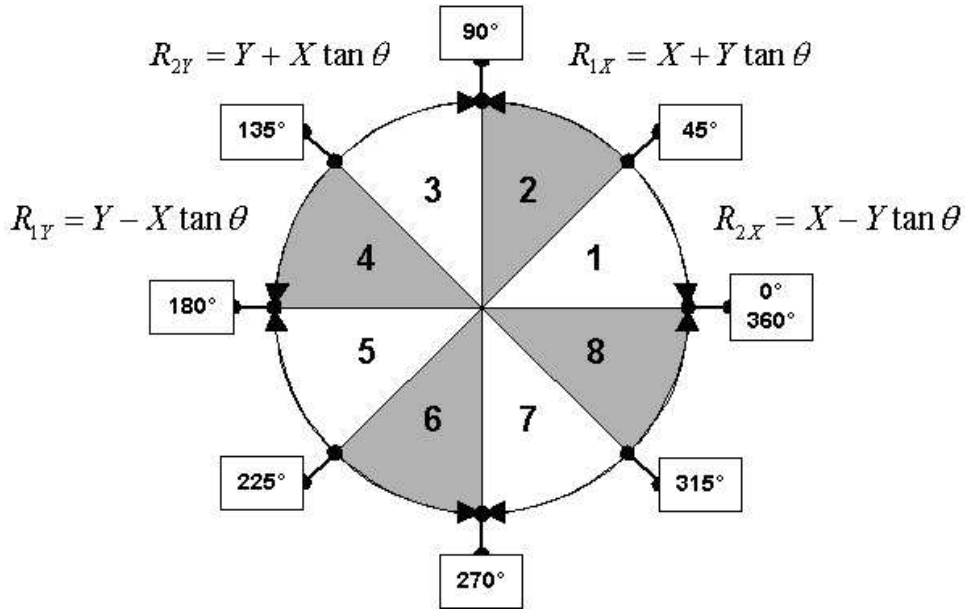


Figure 2: Division into four of the radii search range, with eight regions according to the sign of the calculated radius.

the input to the MST. The voting array is captured in Fig. 6, with the x axis running from 0 to 360° marked off at 45° intervals and the vertical y axis ranging from 0 to 182 pixel radius. The vertical dimension allows for the maximum radius in a 256×256 image. A logarithmic intensity function has been employed so that smaller votes are still visible. Notice the well-defined peaks and the characteristic sinusoidal wave pattern, arising from re-arranging (1) to form:

$$R = (X^2 + Y^2)^{\frac{1}{2}} \sin(\theta + \tan^{-1}(X/Y)) \quad (7)$$

Fig. 7 is the line image created from the voting array. All votes above 32 have been re-created as straight lines. Therefore, it should be stressed that peak detection has *not* been applied to remove shorter edge segments. The plane itself has caused some disturbance to the picture, though the runway borders are apparent. Had the plane itself been first been removed as the result (say) of a template matching algorithm, then Fig. 8 would be the resulting vote array and Fig. 9 is the line image created from the voting array. The main weakness of the image is that the runway leading off to the right is weakly represented.

4 Conclusion

Generation of the voting array is one of the principal obstacles to achieving a real-time general Hough Transform (HT) in hardware. This is because of the need to

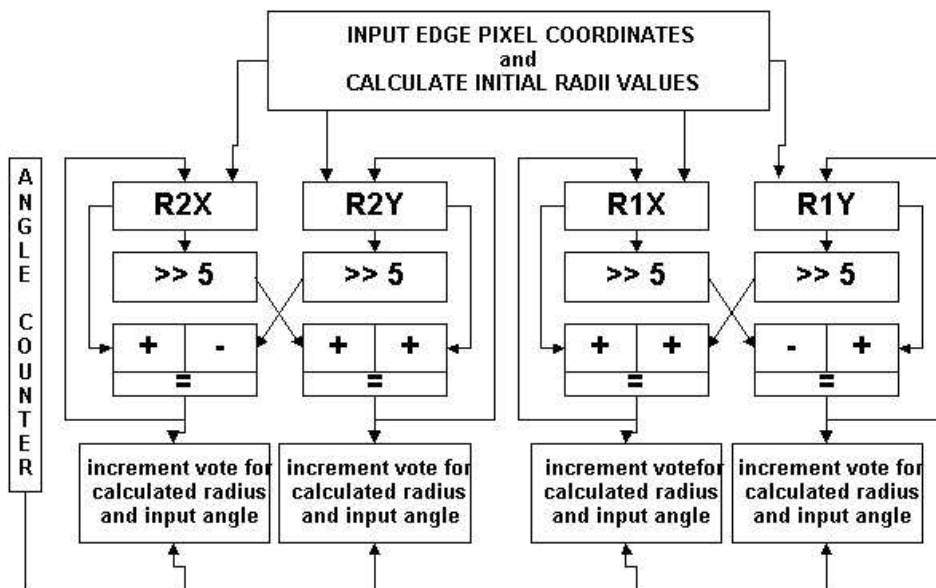


Figure 3: MST cell for simultaneous calculation of four radii values.

calculate trigonometric values. The CORDIC algorithm is one method that has already been applied to the HT on a reconfigurable FPGA. FPGAs are well suited to image-processing applications offering hardware parallelism and fine-grained processing. However, in this paper we consider how the basic CORDIC idea can be modified to improve the throughput through constant-size micro-rotations. The design in this paper used pipelining of edge coordinate inputs and vote generation. Parallelism is also present by simultaneous calculation of pixels within image sectors. As the complexity of the design is much reduced over a pure CORDIC scheme the actual restriction on platform FPGAs is the number of on-chip RAM blocks.

References

- [1] R. Andraka. A survey of CORDIC algorithms for FPGA based computers. In *6th ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA'98)*, pages 191–200, 1998.
- [2] D. Deng and H. ElGindy. High-speed parameterisable Hough transform using reconfigurable hardware. In *Pan-Sydney Area Workshop on Visual Information Processing (VIP2001)*, pages 51–57, 2001.
- [3] S. Wilton, J. Rose, and Z. Vranesic. The memory/logic interface in FPGAs with large embedded memory arrays. *IEEE Transactions on VLSI*, 7(1):80–91, 1999.
- [4] F. Zhou and P. Kornerup. A high speed Hough transform using CORDIC. In *International Conference on Digital Signal Processing (DSP'95)*, 1995.

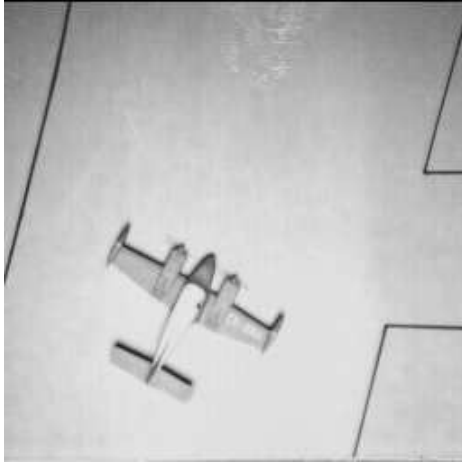


Figure 4: 256×256 "plane" image

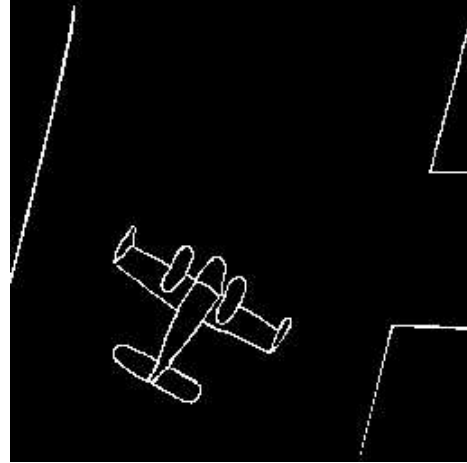


Figure 5: Thresholded edge image

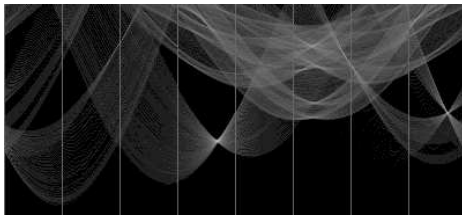


Figure 6: Voting array from Fig. 5

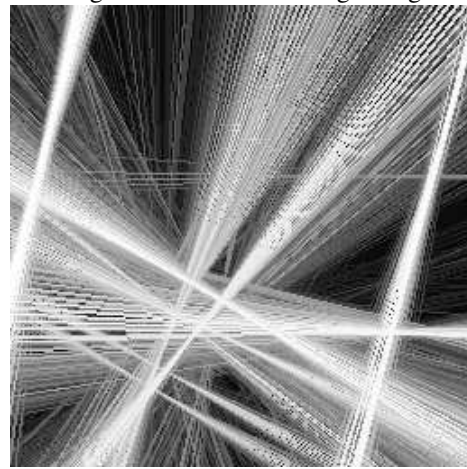


Figure 7: Line image of Fig. 6

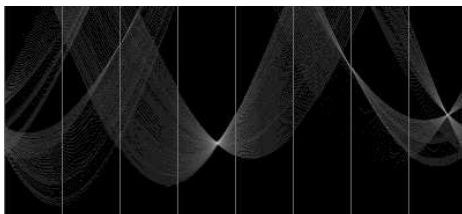


Figure 8: Voting array after removal of the

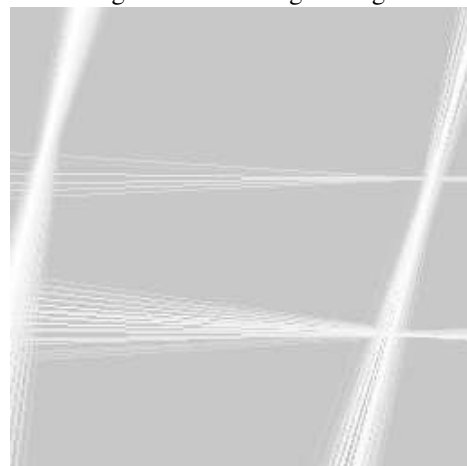


Figure 9: Line image after prior removal of the airplane