17th International Conference on Machine Vision Applications (MVA)
Fully Online, July 25–27, 2021.

Encoding-free Incrementing Hough Transform for High Frame Rate and Ultra-low Delay Straight-line Detection

Ziwei Dong[1], Tingting Hu[1, 2], Ryuji Fuchikami[2], Takeshi Ikenaga[1]

Graduate School of Information, Production and Systems, Waseda University[1]
Kitakyushu, 808-0135, Japan
Panasonic Corporation[2]
Fukuoka, 812-8531, Japan
zwdong@fuji.waseda.jp

Abstract

High frame rate and ultra-low delay straight-line detection plays an increasingly important role in highly automated factories that call for straight-line features to achieve swift locations in real scenes. However, vision systems based on CPU/GPU have a fixed delay between image capture and detection, making straight-line detection challenging to reach an ultra-low delay. Achieving detection nearly simultaneous with capture on the same image is considered. This paper proposes (A) an encoding-free incrementing Hough transform and (B) a partially compressed line parameter space to implement a straight-line detection core on an FPGA board. The encoding-free incrementing Hough transform directly calculates line parameters only by incrementing and initialization while capturing an image. Furthermore, the partially compressed line parameter space reduces the required memory resources and the path delay under the premise of accurate vote recordings for every line feature. The evaluation result shows that the proposals achieve as accurate detection (RMSE of θ on 0.0057, and RMSE of ρ on 2.09) as standard Hough transform (RMSE of θ on 0.0057, and RMSE of ρ on 2.13) and the designed detection core processes VGA (640×480) videos at 1.358 ms/frame delay.

1. Introduction and related works

Straight lines give important geometric information in images. It is one of the most prominent geometric features, especially in human-made objects [1]. To detect straight lines has been required by many visual systems, such as localization [2] and inspection [3] systems. Unlike a straight-line detection system for static images or usual real-time videos (e.g., 60 fps), a high frame rate and ultra-low delay straight-line detection system helps with a swifter and accurate location. However, vision systems based on CPU/GPU have a fixed delay between the whole frame capture and detection, making straight-line detection challenging to reach an ultra-low delay. An ultra-low delay straight-line detection core that works near-simultaneously with capture on the same frame becomes a pivotal solution to minimize the delay.

Standard Hough Transform (SHT) [4] is a widely used method for straight-line detection. The straight-line feature utilized by SHT contains a line's perpendicular distance to the image origin (ρ) and a line's normal direction (θ). Line features are detected by counting the number of edge pixels contribute to them in a line parameter space. Many research works have proved that SHT has the character of highly parallel that is suitable for high-speed implementation on an FPGA board [5][6][7]. Besides, Hough transform involves accumulating votes in memory that is generally not suitable for real-time processing. However, if using a large amount of built-in SRAM of the FPGA in parallel, real-time processing is possible at a fixed rate.

There are many designs for Hough Transform FPGA implementation. For example, an incrementing property of the Hough transform in [8] achieves efficient, simplified perpendicular distance calculation. However, it requires the encoded information acquired from an edge image that becomes an obstacle to reach an ultra-low delay in real scenes. The work [9][10] aim at optimizing line parameter space size. Reducing the size of the line parameter space reaches hardware resource-efficient and the operations in memory space with lower delay. However, the compression in these works brings a higher probability of false detection while reducing required resources.

By taking advantage of the highly parallel character of the Hough transform, this paper targets a high frame rate and ultra-low delay straight-line detection core implemented on an FPGA board. An encoding-free incrementing Hough transform is utilized to remove the extra encoding delay and keep simplified incrementing calculation for perpendicular distances. A partially compressed line parameter space is proposed to reduce the path delay of the line parameter space on hardware.

2. Proposals

This section shows the two proposals for the ultra-low delay straight-line detection core structure. Figure 1 shows the structure. By receiving the grayscale image pixels as input, an edge detector (using the CORDIC algorithm) is firstly utilized to acquire the edge and gradient information in an image. Next, the edge pixel stream output by the edge detector is input to the distance calculation part. Then, the gradient information and the calculated perpendicular distance are input to the line parameter space. The line detection system continuously works by directly processing

the image stream while capturing images and outputs detected line features without an extra delay.

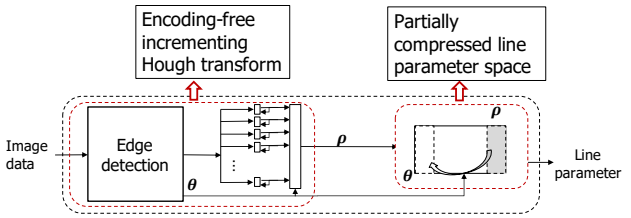

Figure 1. The structure of the straight-line detection core

2.1. Encoding-free incrementing Hough transform

One of the reasons that SHT works slowly to detect lines is a large amount of sequential calculation. In order to speed up the detection, an incrementing property with a run-length encoding of Hough Transform is proposed by Chen et al. [8]. The incrementing property here indicates the perpendicular distances under the same normal direction can be acquired by incrementing operation using equation (2) rather than the original in equation (1). The work [8] implements the commonly used parallel character of Hough transform (the parallel calculation of the perpendicular distances (ρ) under each discrete normal direction (θ) of lines) and the incrementing property with a run-length encoding that improves the calculation efficiency. The incrementing calculation part introduced in [8] receives the pre-encoded image information from edge images that are divided into blocks (area of several neighbor pixels). The merit of the incrementing operation is that it can significantly simplify the calculation in Hough transform to speed up the processing. Besides, the run-length encoder used in this work can skip the non-edge block to acquire more refined edge information to improve the throughput of the incrementing calculation part. Algorithm 1 shows how it processes block by block.

$$\rho_\theta(x,y) = x\cos\theta + y\sin\theta \quad (1)$$

$$\rho_\theta(x+dx, y+dy) = \rho_\theta(x,y) + dx\cos\theta + dy\sin\theta \quad (2)$$

for all encoded block information ***i*** of an image,
  parallel part
  under all normal direction of line θ ($0° \le \theta < 180°$),
    for each edge feature ***j*** in the encoded information
      $\rho_{ij} = \rho_{i0} + \lfloor f_0 + a\cos\theta + b\sin\theta \rfloor$
      Line_Parameter_Space (θ, ρ_{ij}) += 1
    end for
  end parallel part
end for

f_0: fractional part of the incrementing offset
a, b: relative position in the image block

Algorithm 1. Incrementing using encoded information

This method efficiently processes a still image by acquiring the encoded information from the whole image in advance. However, when applied to high-frame-rate video in real scenes, the extra encoding time for every frame causes a delay or extra buffer during processing. Besides, the clock cycles for processing one frame is uncertain by using encoded information, which is unstable for different input frame.

Based on the efficient calculation of the incrementing property, an encoding-free incrementing Hough transform is proposed. It directly processes a pixel stream from a capturing image rather than using the in advance encoded information from image blocks. The edge detector in the design executes the detection on a delivered pixel stream without finishing capturing a whole image. At the same time, the encoding-free calculation directly deals with the edge pixel stream from the edge detector. Before the stream of a new frame comes to the calculation part, the perpendicular distance will be initialized to 0. When the stream comes row by row (or column by column), the perpendicular distance will be incremented $dx\cos\theta$ $(or\ dysin\theta)$. When the next row (or next column) comes, the perpendicular base distance will be incremented $dysin\theta$ $(or\ dxcos\theta)$. Three states in Figure 2 are defined to control the incrementing calculation. Gradient directions acquired from the edge detector reduce the access times of line parameter space and I/O resources. Figure 3 shows the structure comparison between incrementing with encoding and without encoding.

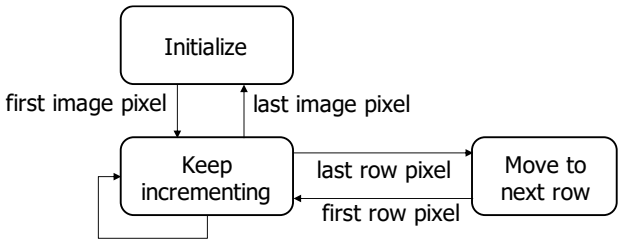

Figure 2. incrementing state controller

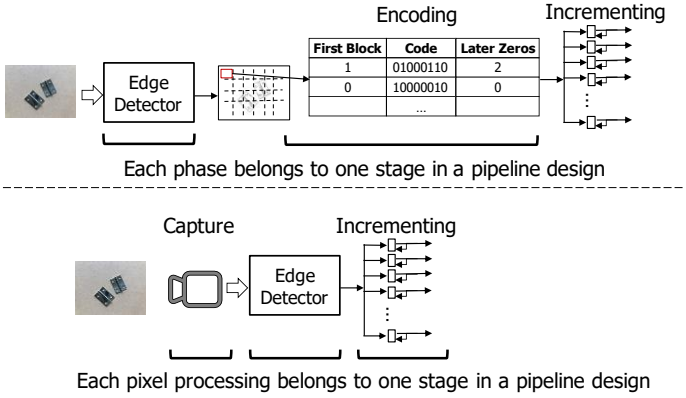

Figure 3. Comparison of incrementing structure with run-length encoding and proposed encoding-free

2.2. Partially Compressed Line Parameter Space

The large line parameter space used by the Hough transform is the main reason for memory resource consumption. From the view of hardware design, a larger size memory

leads to longer path delay. However, to compress a line parameter space should consider the trade-off between false detection and space size. The research work in [10] compresses the line parameter space by overlapping the normal directions. Moreover, according to the observation in [10], compressing the space less than 1/4 of the original one will cause unacceptable false detections. Utilizing this method will save many memory resources. It is acceptable in conventional scenes, which can accept a few false detections. However, for scenes that require no apparent false detection, it will be unsuitable since there is unavoidable false detection caused by the conflicting appearance of different line parameters which share the same cell in the line parameter space.

Inspired by the idea in [10], which reduces memory resources, a partially compressed line parameter space without the conflicting appearance is proposed. Unlike compressing the normal directions of lines to reduce the memory size, the line parameter space is partially compressed by overlapping the perpendicular distances of lines in the parameter space. The idea that perpendicular distances under a normal direction of lines could be overlapped is derived from observing the appearance of line features in images. When a pixel stream from an image is input row by row (or column by column), it can be observed that some line features appear at different periods. For example, in Figure 4, when the pixels comes row by row, the last pixel contributes to the straight-line feature (θ, ρ_1) appears before the first pixel contributes to the feature (θ, ρ_2). Thus, the line features like the example can use the same parameter cell in the line parameter space. The relationships of two-line features that can share the same parameter cell are listed below.

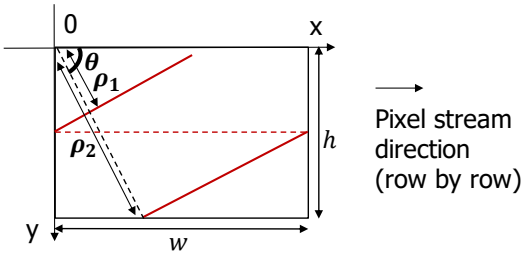

Figure 4. Example of line features that can share the same parameter cell in parameter space

For two lines under the same normal direction in an image:

$$Line1: \rho_1 = \sin\theta\, y_1 + \cos\theta\, x_1$$
$$Line2: \rho_2 = \sin\theta\, y_2 + \cos\theta\, x_2$$

When $0 < \theta \leq \frac{\pi}{2}$ and $0 < \rho_1 \leq \rho_2$, Line1 and Line2 can share the same parameter cell if and only if they appear at different time. While the image is input row by row, it means that the y coordinate of the last pixel contributes to Line1 must be smaller or equal to the y coordinate of the first pixel contributes to Line2 as (3) shows:

$$\frac{\rho_1}{\sin\theta} \leq -\frac{\cos\theta}{\sin\theta} \times w + \frac{\rho_2}{\sin\theta} \quad (3)$$

$$\rho_2 - \rho_1 \geq \cos\theta \times w \quad (4)$$

In the same way, when $\frac{\pi}{2} < \theta < \pi$ and $\rho_1 \leq 0 \leq \rho_2$, these line features can share the same parameter cell if and only if

$$\rho_2 - \rho_1 \geq -\cos\theta \times w \quad (5)$$

Equations (4) and (5) show the relationship the line features should meet to share the same line parameters. The space size to be compressed can be $\sum_{\theta=0}^{\pi-\Delta\theta} \cos\theta \times w$ when image width w is greater than or equal to image height h. In the case of image width w is less than image height h, input image column by column performs the same space compression.

Because the VGA (640×480) video is used to test the designed frame process core and row by row is used. In the design, the relationship $\rho_2 - \rho_1 \geq w$, which meets the conditions in both (4) and (5), is chosen to compress the line parameter space partially. It reduces the required memory address space. Figure 5 shows the partially compressed line parameter space, where $\rho_2 - \rho_1 \geq w$. Using $\rho_2 - \rho_1 \geq w$ compresses the space at most to $1/\sqrt{2}$ of the smallest original size (when image width equals height and image origin is on the center).

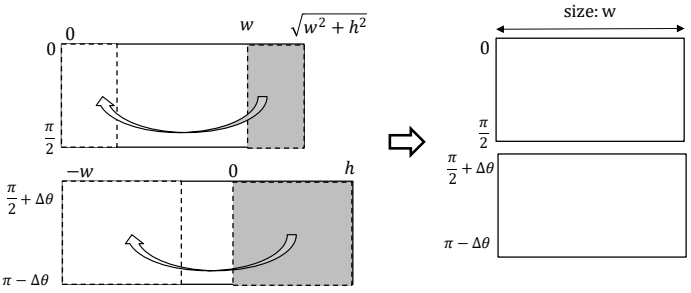

Figure 5. The line parameter space before partial compression and after partial compression

3. Evaluation Result

3.1. Image data

450 straight-line images are utilized to evaluate the accuracy. The images are generated by the line function in OpenCV, including all discrete normal directions ($\theta \in [0, \pi)$, $\Delta\theta = 1°$) of straight lines and selected perpendicular distances ($\Delta\rho = 1$). Gaussian noises and pepper salt noises are added to the images. Each image contains 1 line with a line parameter (ground truth). The line type is the anti-aliased line. Figure 6 shows 2 sample images in the set.

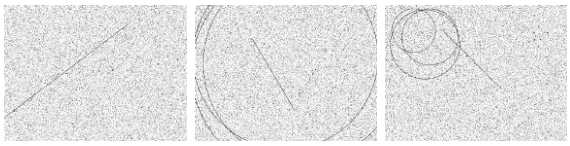

Figure 6. Samples of straight-line images

3.2. Detection accuracy

In order to evaluate the accuracy, the root mean square error (RMSE) $\sqrt{\frac{1}{N}\sum_{i=1}^{N}(result_i - groundtruth_i)^2}$ is measured. The standard Hough transform, the work using the incrementing property with run-length encoding [12], the work proposing compressed line parameter space [14], and the proposed methods in this paper are evaluated. The evaluation results are shown in Table 1. The lower error indicates the higher accuracy. The results show that the proposed methods for high frame rate and ultra-low delay processing can also keep accurate detection as standard Hough transform.

Table 1. The evaluation result of root-mean-square error

Method	error θ (rad)	error ρ (pixel)
Standard HT	0.0057	2.13
Chen et al. VLSI 2011	0.0057	2.35
Northcote et al. ISCAS 2018	0.0119	2.02
proposal 1 + original space	0.0057	2.08
proposal 1 + proposal 2	0.0057	2.09

3.3. Hardware performance and resource usage

The whole structure of the straight-line detection core works on FPGA Xilinx Kintex-7 XC7K325T at the maximum frequency of 226.757MHz. It requires 3.061 μs for a pixel to travel through the straight-line detection core. The processing time for one frame (640 × 480) is 1.358ms.

The resource utilization on the FPGA design is shown in Table 2. The maximum frequency that the implementation can work on is shown in Table 3.

Table 2. Resource usage of the proposed design

	Used	Total	Percentage
# of Slice LUTs	5717	203,800	2.81%
# of Slice registers	6010	407,600	1.47%
# of occupied slices	1653	50,950	0.03%
# of Block RAM Tiles	26	445	5.84%
# of bonded IOBs	62	500	12.40%
# of BUFGCTRLs	1	32	0.03%

Table 3. The maximum frequency of the designed detection core after using partially compressed line parameter space

	proposal 1+ original space	proposal 1 + proposal 2
Maximum frequency (MHz)	209.555	226.757

4. Conclusion

In this paper, an ultra-low delay straight-line detection core based on the Hough transform is implemented. The designed detection core can directly detect straight lines while capturing high-frame-rate real-time grayscale image frames.

This work targets accurate high-speed line detection that can be contributed to visual systems in highly automated factories. Two ideas are proposed to reach ultra-low delay processing. Firstly, an encoding-free incrementing Hough transform is utilized to remove the extra processing delay on video. Secondly, a partially compressed line parameter space is proposed to reduce the path delay on hardware.

As a result, the proposed structure works at 1.358ms delay with the resolution of 640 × 480.

Acknowledgement

This work was supported by KAKENHI (21K11816).

References

[1] J. H. Lee, G. Zhang, J. Lim, and I. H. Suh, "Place recognition using straight lines for vision-based SLAM," *IEEE International Conference on Robotics and Automation*, 2013.

[2] O. A. Aider, P. Hoppenot, and E. Colle, "A model-based method for indoor mobile robot localization using monocular vision and straight-line correspondences," *Robotics and Autonomous Systems*, 2005.

[3] S. Mahadevan, and D. P. Casasent, "Detection of triple junction parameters in microscope images," *Proc. SPIE.* Vol. 4387, 2001.

[4] R. O. Duda, and P. E. Hart, "Use of the Hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol.15.1, pp.11-15, 1972.

[5] M. Nakanishi, and T. Ogura, "A real-time CAM-based Hough transform algorithm and its performance evaluation," *Proceedings of 13th International Conference on Pattern Recognition*, vol.2, pp.516-521, 1996.

[6] X. Zhou, Y. Ito, and K. Nakano, "An efficient implementation of the gradient-based Hough transform using DSP slices and block RAMs on the FPGA," *IEEE International Parallel & Distributed Processing Symposium Workshops*, 2014.

[7] S. M. Karabernou, and F. Terranti, "Real-time FPGA implementation of Hough Transform using gradient and CORDIC algorithm," *Image and Vision Computing*, vol.23.11 pp.1009-1017, 2005.

[8] Z. Chen, A. W. Su, and M. Sun, "Resource-efficient FPGA architecture and implementation of Hough transform," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* vol.20.8, 2011.

[9] P. Ser, and W. Siu, "Memory compression for straight line recognition using the Hough transform," *Pattern recognition letters* vol.16.2, pp.133-145, 1995.

[10] D. Northcote, L. H. Crockett, and P. Murray, "FPGA implementation of a memory-efficient Hough Parameter Space for the detection of lines," *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, 2018