http://doi.org/10.35784/iapgos.2817

# EFFICIENT LINE DETECTION METHOD BASED ON 2D CONVOLUTION FILTER

**Paweł Kowalski, Piotr Tojza**
Gdansk University of Technology, Faculty of Electrical and Control Engineering, Gdańsk, Poland

***Abstract.*** *The article proposes an efficient line detection method using a 2D convolution filter. The proposed method was compared with the Hough transform, the most popular method of straight lines detection. The developed method is suitable for local detection of straight lines with a slope from -45˚ to 45˚. Also, it can be used for curve detection which shape is approximated with the short straight sections. The new method is characterized by a constant computational cost regardless of the number of set pixels. The convolution is performed using the logical conjunction and sum operations. Moreover, design of the developed filter and the method of filtration allows for parallelization. Due to constant computation cost, the new method is suitable for implementation in the hardware structure of real-time image processing systems.*

**Keywords**: image processing, real-time processing, Hough transform, straight lines detection

## EFEKTYWNA METODA WYKRYWANIA LINII Z WYKORZYSTANIEM KONWOLUCYJNEGO FILTRU 2D

***Streszczenie.*** *W artykule zaproponowano efektywną metodę wykrywania prostych z wykorzystaniem dwuwymiarowego filtru konwolucyjnego. Zaproponowana metoda została porównana z transformatą Hough, najpopularniejszą metodą wykrywania linii prostych. Opracowana metoda pozwala na wykrywanie linii prostych o nachyleniu od -45˚ to 45˚. Może również zostać wykorzystana do wykrywania krzywych, których kształt jest aproksymowany za pomocą krótkich prostych odcinków. Zaproponowana metoda charakteryzuje się stałym kosztem obliczeniowym, niezależnym od liczby pikseli. Splot wykonywany jest z wykorzystaniem logicznej koniunkcji oraz sumowania. Ponadto konstrukcja opracowanego filtru oraz zastosowana metoda filtracji pozwala na zrównoleglenie. Ze względu na stały koszt obliczeniowy, zaproponowana metoda nadaje się do implementacji w strukturze sprzętowej systemów przetwarzania obrazu w czasie rzeczywistym.*

**Słowa kluczowe**: przetwarzanie obrazu, przetwarzanie w czasie rzeczywistym, transformacja Hough, wykrywanie prostych

## Introduction

Colinear pixels detection is often used in image processing. For example, for recognizing objects consisting of straight edges. The most commonly used method is the Hough transform [8]. It is characterized by high efficiency and low sensitivity to disturbances, as well as insensitivity to line discontinuity [5, 6]. The main part of this method is the voting procedure, in which each input image pixel is transformed into Hough space. Such a transformation requires determining a set of lines passing through a given pixel. It allows performing some kind of voting in two-dimensional Hough space. The voting results contain information about the number of pixels lying on each straight line. The high efficiency of this method and low noise sensitivity resulted in its modifications for the detection of more complex shapes. For instance, curve or circles detection [11, 12]. This is commonly achieved by extending the Hough space and the voting table to three dimensions. Similarly, the method can be extended to other shapes by broadening the number of dimensions, but this escalates computational cost [1, 8, 11, 12].

Hough transform can also be done by neutral networks, where it also extends the possibilities of detection [4, 7, 10] but due to multiplications during processing it may escalate computational cost.

In another approach, the method itself is not modified but is used for local detections. This approach allows the curve detection if the curve has got a slight curvature and locally is similar to a straight line. The computational cost of the global and local Hough method is similar. Detecting curves requires additional steps to connect the detected straight pieces together. This approach is often less computationally expensive than using the three-dimensional Hough method.

In both cases, broadening the dimensions of the voting matrix or connecting local parts together, allows the detection of more complex shapes, but enlarges the computational cost. The aim of the proposed method is reducing the computational cost compared to the standard Hough transform. The proposed method is dedicated to detect straight sections with an slope from -45˚ to 45˚. The main part of the proposed method is a 2D filtration with a developed set of masks. It is performed using the logical conjunction and sum operations.

## 1. Hough Transform

### 1.1. Cartesian space

Two-dimensional Cartesian Space allows to describe the position of a point by two coefficients ($x$,$y$). The origin of the coordinate system is represented as a point (0,0). By Cartesian space coefficients, it is possible to determine the position of all pixels in an image. A line in Cartesian space image can be represented as set of pixels described by formula (1)

$$y = ax + b\ . \tag{1}$$

Also, the line can be determined by the perpendicular vector $\Psi$ connecting the straight line and the point (0,0). Such a vector has got the constant beginning (0, 0), so $\Psi$ can be described using two coefficients: $\alpha$ and $\rho$, where $\alpha$ is the angle of the vector $\Psi$, and $\rho$ is the length of the vector $\Psi$ (Fig. 1).

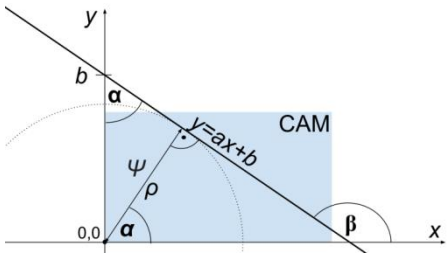


*Fig. 1. Diagram of a line and Ψ vector in Cartesian space*

The relation of the angle $\alpha$ and the coefficient $a$ is defined in the formula (2)

$$a = -ctg(\alpha) \tag{2}$$

The length of the vector $\Psi$ is defined as $\rho$ (3)

$$\rho = sin\alpha \cdot y + cos\alpha \cdot x \tag{3}$$

### 1.2. Hough space

The Hough space is two-dimensional space closely related to the Cartesian space. The line from Cartesian space is established by a single point in Hough space. Such a point is defined as a pair of coefficients $\alpha$ and $\rho$. This representation allows voting in the Hough space.

### 1.3. Voting in the Hough space

The main part of the Hough transformation is the voting procedure, where each pixel in the Cartesian space votes on the lines that pass through it. A single line is defined as a pair $\alpha$ and $\rho$ and calculated by point position *x*, *y* and specified $\alpha$. The set of lines is determined by specifying a set of $\alpha$ coefficients. It is common practice to determine the set of angles $\alpha$ divisible by the selected *step* value. For example, selecting *step* = 1˚, each pixel of the Cartesian space will vote on 360 different lines. Saving the votes is performed by increasing the value of the cell ($\alpha$, $\rho$) of the Hough space matrix. The result of the vote is a two-dimensional Hough space matrix, where the cell position dictates $\alpha$ and $\rho$ and the cell value contains the number of votes.

### 1.4. Inverse transformation

The transformation of the vector ($\alpha$, $\rho$) into the form (1) is called the inverse transformation. In this case, the coefficient $a$ is defined by relation (2), and the coefficient $b$ is determined by relation (4)

$$b = \frac{\rho}{sin\alpha} \tag{4}$$

Using the formula (1) allows determining all the pixels from the Cartesian space lying on the line. It can be used to draw this line in the image, for instance by changing the colour of these pixels.

## 2. New method

The new method was designed for the local detection of straight lines. For this purpose, the image is divided into vertical sections (Fig. 2). Straight lines are detected in each of these sections. Connecting short lines allows to identify straight lines and curves.

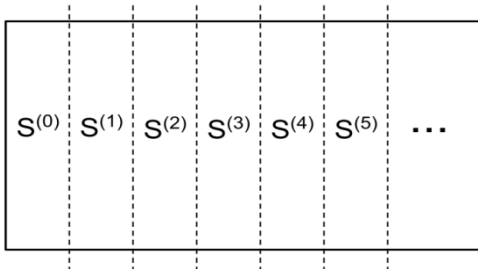


*Fig. 2. The division of the image on the vertical sections*

In the new method, the multiplication of the real numbers was eliminated. There is also no need to determine the values of the trigonometric functions. In the presented method, calculations are performed in the form of a two-dimensional filter using a predefined set of masks. Each mask contains a fragment of one straight line. Convolution of the masks and the window is achieved using logical conjunction. The result is a set of bits. The number of set bits in the string indicates the number of pixels lying on the straight line, which is similar to the results in standard Hough method.

### 2.1. Masks

The masks are designed to detect lines with slope from -45˚ to 45˚. A single mask is expressed by a binary matrix of the size $M_w \times M_h$, where $M_h = 2M_w - 1$. Each mask has been marked with identifiers *id*. The masks are created using (5)

$$m_{x,y}^{id} = \begin{cases} 1 \; if \; y \approx \frac{id - M_w}{M_w - 1} x + M_w - 1 \\ 0 \; else \end{cases} \tag{5}$$

where $m_{x,y}^{id}$ is the cell (*x*,*y*) of the mask $M_h^{id}$. Pixels lying on the line (5) take the value 1, and the rest of them the value 0. A graphical representation of the set of 15 masks of size $8 \times 15$ is shown in Fig. 3. The set bits are represented by white pixels.

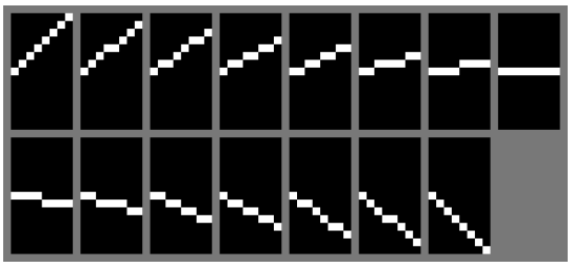

*Fig. 3. Set of 15 masks with the size of 8 × 15*

### 2.2. Convolution

The convolution is performed by moving the window $W$ around the images and convoluting with a set of masks. Image is divided into sections (Fig. 2), where single section can be treated as image $S$ with a width of $M_w$ pixels. The size of the window $W$ and the mask $M$ in the presented algorithm are identical ($M_w \times M_h$). While processing, window $W$ is moving in section $S$ only vertically, so its position can be defined as single variable *j*. Each pixel in window $W^j$ is described by formula (6)

$$w_{x,y}^{j} = s_{x,y+j-M_w+1} \tag{6}$$

where $w_{x,y}^{j}$ is the single pixel of window $W^j$, *j* is the window position, (*x*,*y*) is the position of the pixel in the window and $s_{x,y+j-M_w+1}$ is the pixel $x, y + j - M_w + 1$ of the section *S*.

The convolution with single mask was divided into two steps. First, the intermediate result is obtained by a logical conjunction of binary matrices *M* and *W* (7)

$$C = M \wedge W \tag{7}$$

Then the set bits in *C* are counted. It should be noted that the mask *M* includes only $M_w$ set bits, so the result of the conjunction (7) can be saved in a sequence of $M_w$ bits. The next step is counting set bits in the received binary string. Methods of counting bits was presented in section 2.3. The results from all convolutions are stored in table *R* in form presented in formula (8)

$$r_{j,id} = cb(M^{id} \wedge W^j) \tag{8}$$

where *id* means the mask *id*, *j* the position of the window *W* in section *S*, and *cb* the function of counting set bits in a binary string.

### 2.3. Counting set bits

The most time-consuming part of the new method is counting set bits in the binary string. In the simplest approach, the bits are added sequentially. For the *n* bit sequence it requires a *n*-1 summations.

Another example of a sequential algorithm is the Brian Kernighan algorithm [9]. Execution of this algorithm requires *4·z* elementary operations to determine the number of set bits, where *z* is the number of set bits in the string.

The quickest way is to store all results of counting bits of *n* bits strings in an array called a lookup table. Then the determination of the number of set bits is performed by reading the value directly from the memory. But it may require large amount of memory to keep the lookup table.

Counting the set bits may also be performed using a parallel algorithm. In this approach, all pairs of cells will be summed in each cycle. In each cycle, the number of cells summed is doubled. In that case it requires $\lceil log_2 n \rceil$ cycles.

Another approach is a hybrid method combining techniques with the use of partial results, called lookup table and the simple sequential algorithm. In the hybrid method, the input string is divided into sections of *m* bits. For each section, the number of set bits is determined by simply reading from the lookup table. It takes $\lceil M_w / \lfloor log_2 m \rfloor \rceil$ operations to obtain $\lceil M_w / \lfloor log_2 m \rfloor \rceil$ subsets. Simple addition of these results needs $\lceil M_w / \lfloor log_2 m \rfloor \rceil - 1$ operations. All in all, this procedure

for counting the set bits takes $2 \cdot \lceil M_w / \lfloor log_2 m \rfloor \rceil - 1$ elementary operations. It is faster than simple summation for $m \geq 3$.

Another way to count bits is a combination of a parallel technique and a lookup table. In this case, the string is divided into $m$ bit packets and the partial results are determined by parallel readings from the lookup table. The counting of bits in each section is also performed parallel. In this case, it takes $\lceil log_2 \lceil m/n \rceil \rceil + 1$ cycles and requires $\lceil n/m \rceil$ threads.

## 2.4. Searching for straight lines in the results

The detection of straight lines for both, the standard Hough method and the new method is similar. In both cases, the cell in the result array represents a single straight line, and the value determines the number of pixels lying on that line. Searching for straight lines is usually performed by thresholding the result table [6]. There is slight difference is the size of the resulting arrays of standard Hough method and new method. In the case of Hough, it is $M_h \times \sqrt{{S_h}^2 + {M_h}^2}$ cells, and in new method, it is $M_h \times S_h$.

# 3. Computational cost comparison of the new method and the standard Hough transform

The new method is dedicated for a narrow range of applications. Therefore, both methods are compared in detecting lines with $M_h$ different slopes from -45˚ to 45˚. In the new method, the image $S$ is divided vertically into sections of $M_w$ pixels width (Fig. 2). Each section can be treated as a separate binary image of size $S_h \times M_w$. A similar computation will be performed for each section, so the single section processing cost will be compared. The comparison was divided into three sections, where memory cost, calculation cost and parallel efficiency were compared.

## 3.1. Memory cost

In both algorithms, a significant amount of memory is occupied by the result table and the lookup table. In the case of the standard Hough transform, using a lookup table to store all necessary *sin* and *cos* values reduces the computational cost [2, 3]. The size of such an array is $2M_h$. Additionally, for each section, one voting matrix of the size of $M_h \times \sqrt{{S_h}^2 + {M_h}^2}$ cells is created. The total memory cost is equal to $2M_h + M_h\sqrt{{S_h}^2 + {M_h}^2}$ cells.

In the case of the new method, for the comparison the lookup table for counting set bits in hybrid counting algorithm will be same size as the lookup table of the Hough method, $2M_h$.

The processing results of each section are kept in the matrix of size $M_h \cdot S_h$. The total cost is equal to $2M_h + M_h \cdot S_h$ memory cells. It can be seen that the new approach has got slightly lower memory requirements. The results of the memory cost comparison were shown in Table 1.

*Table 1. Memory cost comparison*

| | Hough | New Method |
|---|---|---|
| lookup table | $2M_h$ | $2M_h$ |
| results table | $M_h \times \sqrt{{S_h}^2 + {M_h}^2}$ | $M_h \cdot S_h$ |
| total cost | $2M_h + M_h \cdot \sqrt{{S_h}^2 + {M_h}^2}$ | $2M_h + M_h \cdot S_h$ |

## 3.2. Calculation cost

The exact calculation cost depends on the platform, which is being used to run the method. For comparative purposes, the impact of platform constraints on the results, such as the time of reading data from memory, was ignored. The computational cost was determined as the number of elementary operations, where each operation such as addition, multiplication or comparison costs one clock cycle.

The standard Hough transform requires three procedures. First, the set bits are searched for. This requires the $M_w \cdot S_h$ comparison operations. For each of the $z$ bits found, a set of $M_h$ straight lines is determined. This means computing a set of $\rho$ coefficients. Each demands two multiplications by the *sin* and *cos* values from the lookup table and single sum operation (3). It means that determining the set of coefficients $\rho$ takes $M_h \cdot z \cdot 3$ elementary operations. Determination of coefficients α is not included in the cost, because they are constant. Each pair of coefficients $\alpha$ and $\rho$ votes by incrementing one cell of the result matrix. This requires the $z \cdot M_h$ increment operations. In total, performing the Hough transformation requires $M_w \cdot S_h + 4M_h \cdot z$ basic operations.

The new method is performed by the convolution of window size $M_w \times M_h$ with the set of $M_h$ masks. After each convolution the window moves one pixel down. Processing the entire image requires $S_h \cdot M_h$ convolutions with masks. Convolution of the window with the single mask requires a single conjunction operation and then the set bits counting. The hybrid algorithm with lookup table of size $2M_h$ performs counting in $2 \cdot \lceil M_w / \lfloor log_2(2M_h) \rfloor \rceil - 1$ elementary operations. So processing the image $S_h \times M_w$ with the new algorithm requires $S_h \cdot M_h \cdot (2 \cdot \lceil M_w / \lfloor log_2(2M_h) \rfloor \rceil - 1)$ basic operations.

Fig. 4 shows the dependence of the number of the set pixels on the computational cost of the processing a single image section of size $32 \times 1080$ pixels. In the standard approach, the number of operations is proportional to the number of set pixels $z$. In contrast, the number of operations for the new method is regardless of the number of set pixels. For the amount of about 6% set pixels, the number of operations of both methods is equal. Above this threshold (6%), the new method requires less operations than the standard method. But below the threshold, the new method requires more operations than the standard method.

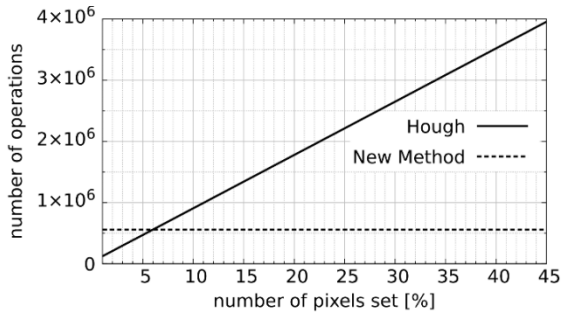


*Fig. 4. The computational cost of processing a 32-pixel wide section of an image*

Fig. 5 indicates the dependence of the image section width $S_w$ on the threshold, where the number of operations of both methods is equal. This threshold is the highest, approximately 6%, for the 20-pixel section width. For sections, which are over 32 pixels wide, the threshold is less than 6%.

It should be noted that the standard method involves the multiplication of the real numbers, and the new algorithm uses only conjunction operations and the addition of integers.

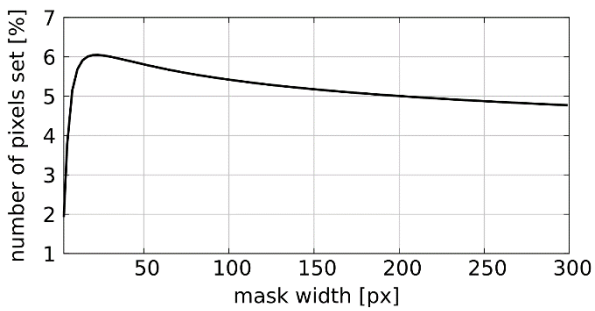


*Fig. 5. Comparison of the computational cost of image processing using the standard Hough transform and the new algorithm*

### 3.3. The cost of parallel processing

The standard approach requires a procedure of set pixels searching, determining $\rho$, and voting. Parallel search for set pixels in one cycle requires $S_w \times M_h$ threads. Each thread performs one comparison operation. Once the $z$ set pixels are found, for each set pixel $M_h$ coefficients $\rho$ are determined. It requires $z \cdot M_h$ threads and 3 clock cycles. Then, each of the threads votes. Multiple threads will modify one cell of the result array, this requires thread synchronization. Even using memory, which allows parallel access to each memory location, the blocking time is equal to $M_h$ operations.

In the case of the new method, the image processing is performed by the convolution procedure. It is composed of the conjunction of windows with masks and counting of set bits. This requires $S_h \cdot M_h$ threads. Each thread will perform a single conjunction operation. Then, the set bits are counted. Parallel counting requires $S_h \times M_h \cdot M_h/2$ threads and $log_2 M_h$ clock cycles. An alternative is the hybrid method, which requires $S_h \cdot M_h$ threads and $2 \cdot \lceil M_w / \lfloor log_2(M_h) \rfloor \rceil - 1$ clock cycles. Each result is stored in a different memory location, so all results can be saved simultaneously. It requires $S_h \cdot M_h$ threads and a single clock cycle. The results of the comparison were depicted in table 2.

*Table 2. Parallel processing cost comparison*

| method | Threads OW | $\frac{operations}{thread}$ | Bloking time |
|---|---|---|---|
| Hough, $z \leq S_h/2$ | $S_h \cdot M_h$ | 5 | $2M_h$ |
| Hough, $z > S_h/2$ | $z \cdot M_h$ | 5 | $M_h \cdot S_h$ |
| New Method, parallel | $\frac{S_h \cdot M_h^2}{2}$ | $\log_2 M_h$ | |
| New Method, hybrid | $S_h \cdot M_h$ | $2 \cdot \left\lceil \frac{M_h}{\lfloor log_2(2M_h) \rfloor} \right\rceil - 1$ | |

With the parallel Hough transform and fully parallel new method, the number of threads varies during processing. Moreover during the voting procedure of parallel Hough transform the threads sleep most of the time. By the contrast, with the new hybrid algorithm, a fixed number of threads is used throughout the entire process, and no one thread is sleeping at any time. Such effective use of computational capabilities makes this method suitable for hardware implementation in a real-time image processing system.

## 4. Comparison of basic operations

### 4.1. Line extension in the next section

The input image is split into sections with the width of $M_w$ (Fig. 2). Lines are detected independently in each section. It makes the detection of the curved lines possible. The shape of a curved line can be approximated as many short straight lines, each in a separate section (Fig. 6).

To obtain the curve, the appropriate short sections must be connected. The convenient approach is to divide the image with sections overlapping by 1 pixel. In this case, the absolute position of the last pixel of the line in section $S^{(i)}$ is the same as the first pixel of the connected line in the section $S^{(i+1)}$. In order to find a connected line, first the position of the rightmost pixel of a given line should be calculated. Then, the straight line from the next section going through that pixel should be found. The cost of this procedure was compared for the results from standard Hough method and the new method.

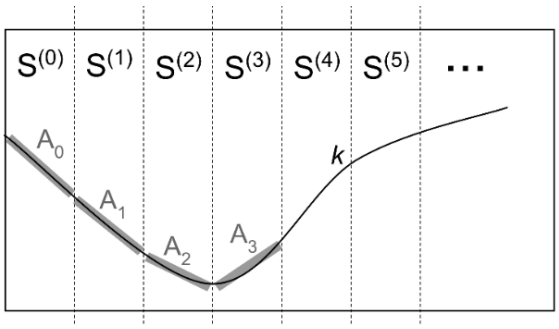


*Fig. 6. Curve in sections*

In the case of the Hough method, each section is associated with the Hough space voting results. The rightmost pixel of a line ($\alpha$,$\rho$) in a section $S^{(i)}$ can be found by inverse transformation and using the formula (1). After locating this pixel in the next section $S^{(i+1)}$, it needs to be transformed to the Hough space getting $M_h$ lines passing through this pixel. Then it should be checked, which of these lines has the most votes in the result table of section $S^{(i+1)}$. Such a procedure requires two operations for the inverse transformation, two operations for determining the point position in the section $S^{(i+1)}$, $3 \cdot M_h$ operations for the Hough transformation, and $M_h$ operations for checking the voting table. In total, it requires $3 \cdot M_h + 4$ basic operations.

In the case of the new algorithm, the results from the section $S^{(i)}$ are stored in the array $R$ of size $S_h \times M_h$. The straight line $r_{y,id}$ means that the line segment starts at the point (0,$y$) of the section and ends at the point ($M_w$,$y - M_w + id$). The continuation of the $r_{y,id}$ line from the $S^{(i)}$ section is $r_{y-M_w+id,k}$ from the $S^{(i+1)}$ section, where $k \in < 0;\ M_h - 1 >$ indicates $M_h$ lines with different slopes. This procedure requires two operations to find the pixel in the next section and $M_h - 1$ operations to find the straight line with the highest score. In total, it requires $M_h + 1$ basic operations.

Straight line extension search for the standard approach requires an $3 \cdot M_h + 4$ operations. However, in the case of the new algorithm, it requires only $M_h + 1$ operations. In this case, the new method is much faster.

### 4.2. Check if a line from the result table passes through the selected point

In the case of the Hough method, the line is described by coefficients $\alpha_L$, $\rho_L$ and in case of new method by $y_L, id_L$. The basic operations is to check if a given point $P = (x_P, y_P)$ from the input image is part of the selected line.

In the case of the standard Hough transform, it consists in the inverse transformation and checking if the pixel $P$ matches the obtained formula (1). This requires two operations for the inverse transformation and three operations to check if the point matches the formula. In total, this requires 5 basic operations.

In case of new method, checking if a line from the result table passes through the selected point requires inserting a pixel $P$ into a blank window. It can be performed by single assignment operation. Then, the conjunction of this window with the mask $id_L$ is made and check if the obtained result contains set bit. This procedure requires one operation to insert a pixel into the window, a single operation of window and mask conjunction, and a single operation to check if the result contains set bit. In total, this requires 3 basic operations. In both cases, the check can be done quickly.

### 4.3. Drawing the line in Cartesian space

After lines detection, drawing found lines in Cartesian space image is the common procedure. In the case of the standard Hough method, the inverse transformation should be performed. Then, for each *x* factor of the image, the *y* factor is calculated using the formula (1), and the $M_w$ pixels are changed to make a line visible. The inverse transformation requires 2 operations. Determining the set of $M_w$ points (*x*,*y*) requires $2 \cdot M_w$ operations. In total, this requires $2 \cdot M_w + 2$ basic operations.

In the case of the new algorithm, masks can be used for drawing. A single line in the voting matrix corresponds to a single mask. Drawing this line can be executed by logical disjunction of the appropriate mask and the appropriate part of the image. It can be done by a single logical operation. It is much faster than drawing a line based on standard Hough transformation results.

### 4.4. Parallel lines search

One of the common procedures of the detected lines processing is searching for parallel lines. In the case of standard Hough transform, lines $L1 = (\alpha_{L1}\rho_{L1})$ and $L2 = (\alpha_{L2}\,\rho_{L2})$ are parallel if $\alpha_{L1} = \alpha_{L2}$. Finding lines in Hough space with identical $\alpha$ coefficient can be done by a single column search of the result matrix. It requires $\sqrt{{S_h}^2 + {M_h}^2}$ comparison operations.

In the case of the new algorithm, The lines $L1 = (y_{L1}, id_{L1})$ and $L2 = (y_{L2}, id_{L2})$ are parallel if $id_{L1} = id_{L2}$. Finding the lines with identical *id* can be done by checking a single column of the result matrix. This requires $S_h$ comparison operations.

In both cases, the search is performed with only comparison operations. The standard method requires $\sqrt{{S_h}^2 + {M_h}^2}$ comparisons, and the new method $S_h$ comparisons.

## 5. Experiments

The developed method was used to detect overhead lines in photos with an environmental background. The test procedure is composed of the Canny edge detection algorithm and the developed method, which divides the image into sections 64 pixels wide. Straight lines are detected in each section. The results of processing sample photos are shown in Fig. 7, 8 and 9. Each example contains three images: a raw image (a), an image after edge detection with the Canny method (b), and an image with straight lines detected by the developed method (c).


*Fig. 7. An example of image processing using the developed method: a) input image, b) Canny edge extraction, c) straight lines extraction by a developed method*


*Fig. 8. Curved wire detection: a) input image, b) Canny edge extraction, c) straight lines extraction by a developed method*


*Fig. 9. Detection of wires in front of trees: a) input image, b) Canny edge extraction, c) straight lines extraction by a developed method*

## 6. Conclusion

The paper presents a new method of collinear pixels detection. The method is suitable for detecting straight lines with a slope from -45˚ to 45˚. An example of the method's application is detecting curves whose shape is locally approximated using short straight lines. The proposed method is characterized by a constant computational cost regardless of the number of set pixels. It is achieved by two-dimensional filtering using a dedicated set of masks performed using conjunction and sum operations. It eliminates real numbers multiplication. Due to constant computation cost, the new method is suitable for hardware implementation in a real-time processing system. The computational cost of selected operations performed after detecting straight lines was also compared. It has been shown that using the results of the new algorithm, it is faster to perform operations such as:

- line extension in the next section,
- checking whether a line passes through a selected point,
- drawing a line in the Cartesian space,
- searching for parallel lines.

The comparison was presented in table 3.

*Table 3. Comparison of calculation costs of selected procedures after detecting straight lines*

| procedure | Hough | New Method |
|---|---|---|
| Line extension | $3 \cdot M_h + 4$ | $M_h + 1$ |
| Checking if a line passes through the point | 5 | 3 |
| Drawing a line in Cartesian space | $2 \cdot M_h + 2$ | 1 |
| Parallel line search | $\sqrt{{S_h}^2 + {M_h}^2}$ | $S_h$ |

## References

[1] Ballard D. H.: Generalizing the hough transform to detect arbitrary shapes. Pattern recognition 13(2), 1981, 111–122.
[2] Elhossini A., Moussa M.: Memory efficient FPGA implementation of hough transform for line and circle detection. 25th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), 2012, 1–5
[3] Guan J., An F., Zhang X., Chen L., Mattausch H. J.: Real-time straight-line detection for xga-size videos by hough transform with parallelized voting procedures. Sensors 17(2), 2017, 270.
[4] Han Q., Zhao K., Xu J., Cheng M. M.: Deep hough transform for semantic line detection. 2020, arXiv preprint arXiv:2003.04676.
[5] Illingworth J., Kittler J.: A survey of the hough trans form. Computer vision, graphics, and image processing 44(1), 1988, 87–116.
[6] Kowalski P., Smyk R.: Straight lines detection in digital image using hough transform. Zeszyty Naukowe Wydziału Elektrotechniki i Automatyki Politechniki Gdanskiej 61, 2018, 45–48.
[7] Milletari F., Ahmadi S. A., Kroll C., Plate A., Rozanski V., Maiostre J., Levin J., Dietrich O., Ertl-Wagner B., Bötzel K., et al.: Hough-cnn: deep learning for segmentation of deep brain regions in mri and ultrasound. Computer Vision and Image Understanding 164, 2017, 92–102.
[8] Mukhopadhyay P., Chaudhuri B. B.: A survey of hough transform. Pattern Recognition 48(3), 2015, 993–1010.
[9] Ritchie D. M., Kernighan W., Lesk M. E.: The C programming language. Prentice Hall Englewood Cliffs, 1988.
[10] Serra P. L., Masotti P. H., Rocha M. S., de Andrade D. A., Torres W. M., de Mesquita R. N.: Two-phase flow void fraction estimation based on bubble image segmentation using randomized hough transform with neural network (rhtn). Progress in Nuclear Energy 118, 2020, 103133.
[11] Shehata Hassanein A., Mohammad S., Sameer M., Ehab Ragab M.: A survey on hough transform, theory, techniques and applications. 2015, arXiv:1502.02160.
[12] Ye H., Shang G., Wang L., Zheng M.: A new method based on hough transform for quick line and circle detection. IEEE 8th International Conference on Biomedical Engineering and Informatics (BMEI), 2015, 52–56.

**Ph.D. Eng. Paweł Kowalski**
e-mail: pawel.kowalski@pg.edu.pl

Assistant professor at the Department of Electrical and High Voltage Engineering, Faculty of Electrical and Control Engineering, Gdańsk University of Technology. His research interest include computer vision, real time image processing and embedded systems and hardware structures in FPGA.

http://orcid.org/0000-0002-0913-1408

**Ph.D. Eng. Piotr M. Tojza**
e-mail: piotr.tojza@pg.edu.pl

Piotr Tojza finished his M.Sc. in 2011 at Gdansk University of Technology in the field of Control Engineering and Robotics. In 2016 he obtained his Ph.D. in Bioengineering. His main field of scientific interests are: biomechatronics, medical diagnostic systems and decision supporting systems. From 2016 he is an adjunct professor at Gdansk University of Technology, Department of Biomechatronics. From 2017 he is also a medical student at the Medical University of Gdansk, Faculty of Medicine.

http://orcid.org/0000-0002-0837-0976

*otrzymano/received: 10.11.2021* *przyjęto do druku/accepted: 15.12.2021*