Straight-Line Recognition in a Virtual Hexagonal Grid Using Hough Transform

Moïse Ouedraogo(✉) and Abdoulaye Sere

Equipe de Recherche Signal, Image et Communications (ER-SIC), Laboratoire d'Algèbre de Mathématiques Discrètes et Informatique (LAMDI), Université Nazi BONI (UNB), Bobo-Dioulasso, Burkina Faso
moisewedra@gmail.com, abdoulaye.sere@recifaso.org

Abstract. The Hough transform is a powerful mathematical tool designed for detecting geometric patterns, including straight lines, circles, and other shapes, within images. The fundamental idea behind this transformation is to represent the points of an image in a parameter space specific to the pattern one aims to detect. This paper focuses on analytical straight-line recognition, using standard Hough Transform in a virtual grid. Each cell in the virtual grid is hexagonal. The method consists of superimposing a virtual grid on the image and calculating the transformation of lit pixels in hexagonal cells where the rate of lit pixels exceeds a threshold. The voting threshold is a criterion that determines how many votes are required for a line to be considered detected. It is an important parameter that influences the quality and reliability of line detection in an image. A lower voting threshold includes noise and consequently degrades the detection quality. Our approach takes advantage of this limitation to optimize analytical straight-line recognition. The experimental results show an improvement in execution time compared with the standard Hough transform method.

Keywords: Hough Transform · Reconstruction · straight-line Recognition · Virtual Hexagonal Grid

1 Introduction

The Hough transform is a robust technique in computer vision, serving to identify shapes and patterns within images with remarkable accuracy. Its significance lies in its ability to transcend the confines of pixel-based analysis, delving instead into the realm of parameter space exploration. This approach was originally conceived by Paul Hough [3] in 1962, with the aim of detecting straight lines in noisy images. Since its inception, the Hough transform has evolved and generalized, making it adaptable to many applications. Notably, it has been extended to recognize arbitrary shapes [4,40,54], intricate ellipses [5], curves [41,45,47,53], circular formations [6,50], and analytically defined lines [7,44,55].

Over the years, the researchers have developed several variants [4,22–38] of Hough transform. It has been applied in different fields, including detecting roads in satellite

K. Cheikh M. F. Kebe et al. (Eds.): InterSol 2024, LNICST 610, pp. 197–213, 2025.
https://doi.org/10.1007/978-3-031-86493-3_16

images [8,9], managing traffic by determining road saturation in [10], lane tracking [51], agriculture [42,48,52], breeding [56],building [49], chirology [43], detecting lineaments in satellite images, autonomous vehicles [11,46,57], breast cancer detection screening [39], and localizing robots [12]. To improve computation time, Rectangular Hough Transform and Triangular Hough Transform were proposed by Traore [13] and Moïse Ouedraogo [14], respectively. These methods involve meshing the image with a rectangular or triangular grid and then applying the standard Hough transform.

Another technique to improve computation time is Parallel Hough Transform on a triangular grid by Abdoulaye Sere and al [15]. Sere and others in [16] proposed an Hough Transform method based on the map-reduce algorithm to perform speedily Big Data of images for pattern recognition. These works have been also extended by Mateus Coelho and others in [6] to propose circle recognition based on the map-reduce algorithm.

In this document, we focus on hexagonal image meshing. The use of a hexagonal grid offers significant advantages in the field of image processing. Drawing parallels with the biological domain, the complex hexagonal pattern observed in the retina of the human eye, as discussed in detail by Curcio et al. [17], testifies to the potential inherent in this geometric arrangement. This natural hexagonal arrangement of sensory cells aligns with the principles of efficiency and optimisation, inspiring its integration into image processing methodologies. The gains in efficiency brought about by the hexagonal grid go beyond the biological domain. Asharindavida et al. [18] claim that computational operations performed on images presented in a hexagonal format exhibit accelerated processing speeds. This unequivocally contributes to a notable enhancement in the overall efficiency of image processing tasks. The ramifications of this acceleration reverberate across diverse applications, ranging from cutting-edge deep learning techniques such as convolutional neural networks [19], to intricate geospatial computing endeavors [20], and even to foundational image pre-processing steps like edge detection [21]. The adoption of a hexagonal grid in image processing thus ushers in a new dimension of expedited operations and streamlined computational efforts. The implications are far-reaching, offering the potential to reshape the landscape of various domains reliant on efficient and rapid image analysis. As technological advancements continue to unfold, the hexagonal grid stands as a testament to the interplay between natural patterns and computational innovation, culminating in an augmented capacity to decipher the intricacies of visual data.

The central question underlying this study concerns the application of the Hough transform on a virtual hexagonal grid superimposed on an image. How can the Hough transform be adapted to ensure maximum accuracy and efficiency in the detection and recognition of straight lines using virtual hexagonal grids?

To meet this challenge, we need to explore the adjustments needed to optimise the exploitation of the geometric features inherent in the grid. This paves the way for significant progress in the detection of linear structures using virtual hexagonal grids.

The main objective of this study is to optimise the recognition of straight lines using the Hough transform technique by integrating a virtual hexagonal grid into the images. This objective encompasses several sub-objectives. Firstly, the research involves the design and superimposition of a virtual hexagonal grid on the image. Secondly, the

study looks at the complex task of adapting the Hough transform in the specific context of a virtual hexagonal grid. By addressing these objectives, the research aims to advance the efficiency of straight line recognition in computer vision, providing valuable insights into the incorporation of virtual grids in digital image analysis.

This paper is organized as follows: Sect. 2 recalls basic definitions related to the analytical straight line and standard Hough transform to aid in understanding the following sections. Section 3 describes the method using various algorithms, while Sect. 4 presents experimental results of previous algorithms.

2 Preliminaries

A digital image is made up of a set of discrete elements, called pixels, arranged in a two-dimensional space. It can be considered as a discrete set in which the mathematical principles of discrete geometry can be applied.

Definition 1 (Analytical straight line [1]). *Analytical straight line with parameters* (a,b,μ) *and thickness* w *is defined by the set of integer points* (x,y) *verifying:*

$$\mu \leq ax+by < \mu+w\ ,\ (a,b,\mu,w) \in \mathbb{Z}^4,\ gcd(a,b)=1 \tag{1}$$

Analytical straight line is:

- thin, if $w < \max(|a|,|b|)$
- naif, if $w = \max(|a|,|b|)$
- standard, if $w = (|a|+|b|)$
- thick, if $w > (|a|+|b|)$

Definition 2 (Standard Hough Transform (SHT) [2]). *Let* $\mathscr{I} \subset \mathbb{R}^2$ *be an image space. Let* l *be the number of columns in an image. Let* h *be the number of rows in an image. Suppose that the point* $(x,y) \in \mathscr{I}$*. The dual of* (x,y)*, denoted* $S(x,y)$ *is the Standard Hough Transform defined by the set of continuous points :*

$$\{(\theta,r) \in [0,\pi] \times [-\sqrt{l^2+h^2}, \sqrt{l^2+h^2}] /\ r = x\cos\theta + y\sin\theta\} \tag{2}$$

Definition 3 (Hexagon). *A hexagon is a six-sided polygon.*
γ *define the size of hexagon.*
AB, BC, CD, DE, EF and FA are the sides of hexagon illustrated in Fig. 1.

The hexagon is subdivided into three parts as illustrated in Fig. 1(a):

- the isosceles triangle ABF
- the rectangle BCEF
- the isosceles triangle BDE

Thales's theorem is used to browse pixels in the isosceles triangular area of each hexagon. The theorem is presented as follows:

Theorem 1 (Thales). *Let ABG be a right triangle. Let N and P be points on lines (AB) and (AG) respectively, with line NP parallel to line (BG). Then:* $\frac{AP}{AG} = \frac{NP}{BG} = \frac{AN}{AB}$.

Thales's theorem shows proportionality between parallel lines. For example, using the equation $(\frac{AP}{AG} = \frac{NP}{BG})$ from the theorem, we can find that $NP = \frac{AP \times BG}{AG}$. If $BG = AG = \gamma$, then $NP = AP$. Thus, $NP = PK$ because ABF is an isosceles triangle. Similarly, we have $MQ = QR = QD$ as shown in Fig. 1(b).

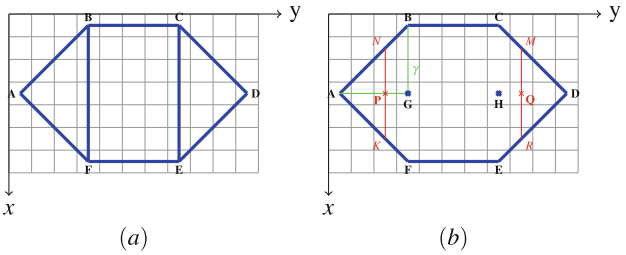


Fig. 1. Hexagon ABCDEF

3 Method Description

This section explains our method which involves placing a virtual hexagonal grid on the binary image and computing the rate of lit pixels in each cell. The dual of a hexagonal cell is computed when its lit pixel rate exceeds a reference rate. The hexagonal Hough transform method is applicable to binary images. Depending on the type of image (classique, multiband, RADAR), the corresponding pre-processing must be applied to obtain a binary image where the objects of interest are contrasted against the background.

3.1 Virtual Hexagonal Grid

Designing a virtual hexagonal grid using the image space mesh involves dividing an image into hexagonal cells and positioning these cells in such a way that they cover the image consistently. In this paper, the virtual hexagonal grid consists of hexagonal cells with the same characteristics. It is therefore a regular mesh(Illustrated in Fig. 2(a)).

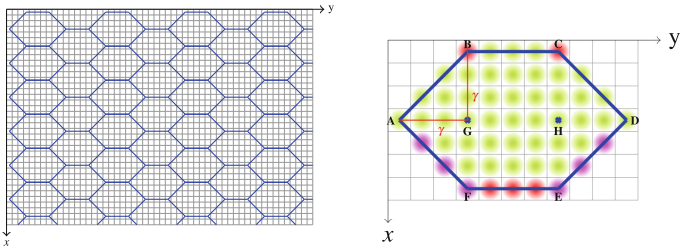
```
Function mesh(image, γ : integer) : table of integer
    %γ is the size of hexagonal mesh;
    Variables: tab : table of 3 integers ;
    Output: tab : table of 3 integers;
    Begin
        Nl ← the height of the image in pixels;
        Nc ← the width of the image in pixels ;
        tab[0] ← γ;
        tab[1] ← Nl mod(2γ−2);% residues on the height
        tab[2] ← Nc mod(4γ−2);% residues on the width
        Return: tab;
    end
end
```

Algorithm 1: Hexagonal mesh function

The Algorithm 1 is used for virtual hexagonal grid conception. With input parameters (γ and image) the Algorithm 1 returns an array of 3 integers. The first value in the array is γ expressed in pixels and noted *px*, the second and third values are the row residuals and column residuals, respectively.

3.2 Algorithm for Hexagonal Selection

The criteria for selecting a hexagon from the hexagon grid is the lit pixel rate in the hexagon. This rate is calculated with the Algorithm 4. It should be noted that for a



(a) Hexagonal grid (b) Area of the hexagon ABCDEF

Fig. 2. Hexagonal mesh

given hexagon of the grid, all the pixels are not considered in the calculation of the rate. In Fig. 2(b) the pixels of green color are those considered in the computation. The others will be taken into account in the neighboring hexagons, as illustrated in Fig. 3. The exclusion of the red and magenta color pixels (with Algorithm 2) makes it possible not to consider twice the same pixel in two different hexagons.

```
Function Relate(A, B : tables of two integers) : list
    Variables: MAT :an integer couple list;
    Begin
        MAT ← points (pixels) coordinates
        between A and B, A and B excluded ;
        Return: MAT;
    end
end
```

Algorithm 2: Relate pixels between A and B

3.3 Analytical Straight Line Recognition in a Virtual Hexagonal Mesh

The Algorithm 4, is the complete algorithm proposed for straight-line recognition in a hexagonal grid. Threshold, α, and γ are the input parameters of the Algorithm 4. The duals of the selected hexagons are computed with Algorithm 3 in the accumulator to determine the highest votes.

```
Procedure Acc_update(): void
    % Updating the accumulator
    if 0 ≤ i ≤ Nl and 0 ≤ j ≤ Nc then
        if img[i, j] ≠ 0 then
            for itheta between 0 and accum_row do
                theta ← itheta × dtheta;
                rho ← j × cos(theta) + i × sin(theta);
                irho ← int(rho);
                if irho > 0 and irho < accum_column then
                    accum[itheta][irho] ← accum[itheta][irho] + 1
                end if
            end for
        end if
    end if
End
```

Algorithm 3: Updating the accumulator data.

```
Function Hexagonal(imge,γ,α,threshold):table
    Pre-condition: 2 ≤ γ ≤ min( (Nl+2)/2 ; (Nc+2)/2 ), 0 ≤ α ≤ 1, 0<threshold
    Variables: tab : table of 6 integers ; A, B, C,D, E, F, G, H : tables of two integers;
    accum_ligne, accum_colon, irho, Nl, Nc : integers; accum : accumulator ; Seuil, α, dtheta, drho, rho,
    theta, DE : reels;
    Begin
        Nl ← number of rows of image; Nc ← number of columns of image; % the dimensions of the image ;
        α = 0.8 % The rate of lit pixels from which the hexagon is selected ; threshold ← 150 % the minimum
          number of votes ;
        accum_row ← 180 ; accum_column ← E(√(Nc² + Nl²)); % the dimensions of the matrix ''accum'';
        γ ← 3; dtheta = π/180 ; tab ← mesh(image, γ) ; Ht ← Nl − tab[3]; La ← Nc − tab[4]; H1 ← Ht + 2γ − 2; L1 ← La + 4γ − 2;
        for 3γ − 2 ≤ y ≤ L1 with the step of 4γ − 2 do
            for 2γ − 2 ≤ x ≤ H1 with the step of 2γ − 2 do
                A[0] = x − tab[0] + 1 ; A[1] = y − 3tab[0] + 2 ; B[0] = x − 2tab[0] + 2 ; B[1] = y − 2tab[0] + 1 ; C[0] = x − 2tab[0] + 2 ;
                C[1] = y − tab[0] + 1 ; D[0] = x − tab[0] + 1 ; D[1] = y ; E[0] = x ; E[1] = y − tab[0] + 1 ; F[0] = x ;
                F[1] = y − 2tab[0] + 1 ; G[0] = x − tab[0] + 1 ; G[1] = y − 2tab[0] + 1 ; H[0] = x − tab[0] + 1 ; H[1] = y − tab[0] + 1 ;
                if count(img, A,B, D, E, F, G, H)≥ α then
                    MAT1 ← Relate(A, F); MAT2 ← Relate(F,E); MAT3 ← Relate(E,D);
                    for A[1] ≤ j ≤ D[1] do
                        if A[1] ≤ j ≤ G[1] − 1 then
                            NP ← |j − A[1]|
                            for A[0] − NP ≤ i ≤ A[0] + NP do
                                if (i,j) is an element of MAT1 then
                                    do nothing;
                                else
                                    Acc_update();
                                end if
                            end for
                        end if
                        if G[1] ≤ j ≤ H[1] then
                            for B[0] ≤ i ≤ F[0] do
                                if (i,j)is an element of MAT2 then
                                    do nothing;
                                else
                                    Acc_update()
                                end if
                                ;
                            end for
                        end if
                        if H[1] ≤ j ≤ D[1] then
                            MQ ← |j − D[1]|
                            for A[0] − MQ ≤ i ≤ A[0] + MQ do
                                if (i,j)is an element of MAT3 then
                                    do nothing;
                                else
                                    Acc_update()
                                end if
                                ;
                            end for
                        end if
                    end for
                end if
            end for
        end for
        for γ − 1 ≤ y ≤ L1 with the step of 4γ − 2 do
            for γ − 1 ≤ x ≤ H1 with the step of 2γ − 2 do
                A[0] = x − tab[0] + 1 ; A[1] = y − 3tab[0] + 2 ; B[0] = x − 2tab[0] + 2 ; B[1] = y − 2tab[0] + 1 ; C[0] = x − 2tab[0] + 2 ;
                C[1] = y − tab[0] + 1 ; D[0] = x − tab[0] + 1 ; D[1] = y ; E[0] = x ; E[1] = y − tab[0] + 1 ; F[0] = x ;
                F[1] = y − 2tab[0] + 1 ; G[0] = x − tab[0] + 1 ; G[1] = y − 2tab[0] + 1 ; H[0] = x − tab[0] + 1 ; H[1] = y − tab[0] + 1 ;
                if count(img, A,B, C, D, E, F, G, H)≥ α then
                    MAT1 ← Relate(A, F); MAT2 ← Relate(F,E); MAT3 ← Relate(E,D);
                    for A[1] ≤ j ≤ D[1] do
                        if A[1] ≤ j ≤ G[1] − 1 then
                            NP ← |j − A[1]|
                            for A[0] − NP ≤ i ≤ A[0] + NP do
                                if (i,j) is an element of MAT1 then
                                    do nothing;
                                else
                                    Acc_update()
                                end if
                                ;
                            end for
                        end if
                        if G[1] ≤ j ≤ H[1] then
                            for B[0] ≤ i ≤ F[0] do
                                if (i,j) is an element of MAT2 then
                                    do nothing;
                                else
                                    Acc_update();
                                end if
                            end for
                        end if
                        if H[1] ≤ j ≤ D[1] then
                            MQ ← |j − D[1]|
                            for A[0] − MQ ≤ i ≤ A[0] + MQ do
                                if (i,j) is an element of MAT3 then
                                    do nothing;
                                else
                                    Acc_update()
                                end if
                                ;
                            end for
                        end if
                    end for
                end if
            end for
        end for
        Search for maxima in the accumulator ;
        Line drawing ;
    end
end
```

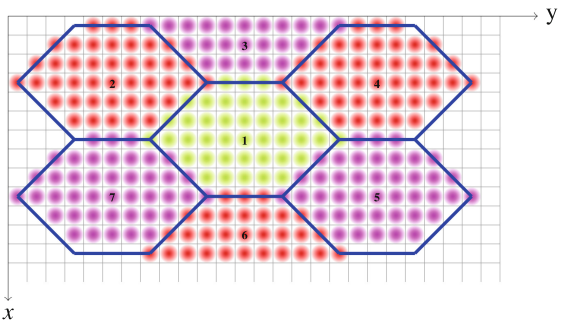


Fig. 3. A selection of hexagonal cells

3.4 Complexity

The Algorithm 4 for the hexagonal Hough transform calls several functions:

- the *mesh* function (Algorithm 1) with complexity $O(1)$.
- the *count* function (Algorithm 4) with complexity $O(S_H) = O(4(\gamma-1)(\gamma-\frac{1}{2})) = O(\gamma^2)$.
- the *Relate* function (Algorithm 2) with complexity $O(1)$.
- the *Acc_update* function (Algorithm 3) with complexity $O(1)$.

The Algorithm 4 of the Hexagonal Hough Transform begins with the initialization of variables and the array *tab*, as well as the calculation of the image and accumulator dimensions, all of which have a constant cost, hence $O(1)$. Then, it calls the *mesh* function whose complexity is $O(1)$.

Inside the main loops, there are two nested loops iterating over the points inside each hexagon, inside which are constant cost operations and the *Acc_update* function of complexity $O(1)$. The complexity of these loops is therefore $O(4(\gamma-1)(\gamma-\frac{1}{2})) = O(\gamma^2)$.

Thus, the cost of the two main nested loops traversing the hexagons of the grid, whose total number of iterations depends on the size of the image and the size of the hexagonal cells, containing a block of assignment operations of constant cost, a condition test on the *count_hex* function of complexity $O(\gamma^2)$, three assignments calling the *Rely* function of constant cost, the two nested loops of cost $4(\gamma-1)(\gamma-\frac{1}{2})$ traversing the pixels inside the hexagon, is $Nl/(4\gamma-2) \times Nc/(2\gamma-2) \times (1+4(\gamma-1)(\gamma-\frac{1}{2}) + 1+4(\gamma-1)(\gamma-\frac{1}{2}))$, therefore the complexity is $O(Nl/(4\gamma-2) \times Nc/(2\gamma-2) \times (1+4(\gamma-1)(\gamma-\frac{1}{2}) + 1+4(\gamma-1)(\gamma-\frac{1}{2}))) = O(Nl \times Nc)$.

Considering these points, the total complexity of the Hexagonal Hough Transform Algorithm 4 algorithm is $O(Nl \times Nc)$, where Nl and Nc are the image dimensions. This corresponds to the complexity of the standard Hough transform.

4 Simulation and Discussions

The Python programming language was utilized to implement the preceding algorithms. In order to carry out the simulation, a computer with the following specifications was used:

- Processor: Intel(R) Core(TM) i5 CPU M 480 @ 2.67 GHz 2.67 GHz
- Ram memory: 3.79 Go
- Operating System: kali linux, 64 bits.

Lemma 1. *Let γ be the size of an hexagon. The hexagon's surface S_H, represented by the green part of Fig. 2(b), is calculated using the following formula:*

$$S_H = 4(\gamma-1)(\gamma-\frac{1}{2}) \tag{3}$$

Proof. (Lemma 1). Consider γ the size of the Hexagon.

- calculation of the surface S_r of the rectangle in which the hexagon is inscribed: $S_r = L \times l = (3\gamma-1)(2\gamma-1)$ where $L = (3\gamma-1)$ and $l = (2\gamma-1)$ represent, respectively the length and the width of the rectangle.
- calculation of the surface S_t of the four right-angled triangles (at the vertices of the rectangle): $S_t = 4(\frac{\gamma^2-\gamma}{2})$
- calculation of the surface S_e of the set of excluded pixels: $S_e = (\gamma-1)+2\times(\gamma-1)+2 = 3\gamma-1$ where$(\gamma-1)$ represents the number of red pixels at the bottom of the Hexagon, $2\times(\gamma-1)$ represents the number of magenta color pixels on the left and right sides of the hexagon and end, 2 represents the two red pixels at the top of the Hexagon.
- calculation of the surface of the hexagon (only the green part of Fig. 2(right)): $S_H = S_r - (S_t + S_e)$.
  Which gives after simplification and factorization, $S_H = 4(\gamma-1)(\gamma-\frac{1}{2})$ □

For instance, for $\gamma = 4$, $S_H = 4(4-1)(4-\frac{1}{2}) = 4\times 3\times 3.5 = 42\ px^2$ as shown in the Fig. 2(b). The unit of measurement for surface area is in pixels squared (px^2).

We simulated these algorithms on a building image presented in Fig. 5(a) and on a road image presented in Fig. 6(a). On this classique image, pre-processing was applied with the Canny filter (threshold between 500 and 200) to get edges. The resulting images on Fig. 5(b) and Fig. 6(b) are binaries. Algorithm 4, is applied to the resulting binary image to recognize straight lines. In all the following, the detected digital lines on images are highlighted in green.

4.1 The Case of a Building Image

We vary the surface of the grid hexagons as well as the rate of lit pixels in the hexagons.

The number of detected lines and the execution time of Algorithm 4, are summarized in the Table 3, in appendix.

Analysis of these results shows that if the hexagon surface is increasing, the execution time decreases in most of the cases, as illustrated on Fig. 4(right). Likewise, if the rate of light pixels is increasing, the execution time decreases in most cases, as illustrated on Fig. 4(left). As for the number of lines detected, it changes with the variation of the surface of the hexagon and decreases when the Rate α increases.

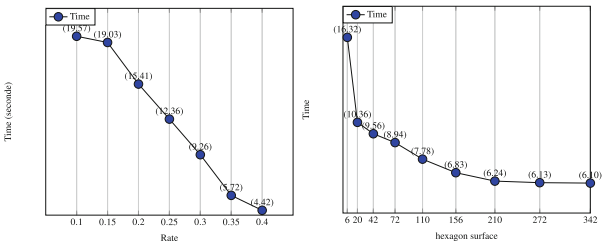


Fig. 4. Building's case: (left) execution time varies according to the rate and the parameters (threshold = 200, $\gamma = 4$); (right) Execution time varies according to the hexagon surface with the parameters (threshold= 200, $\alpha = 0.3$)

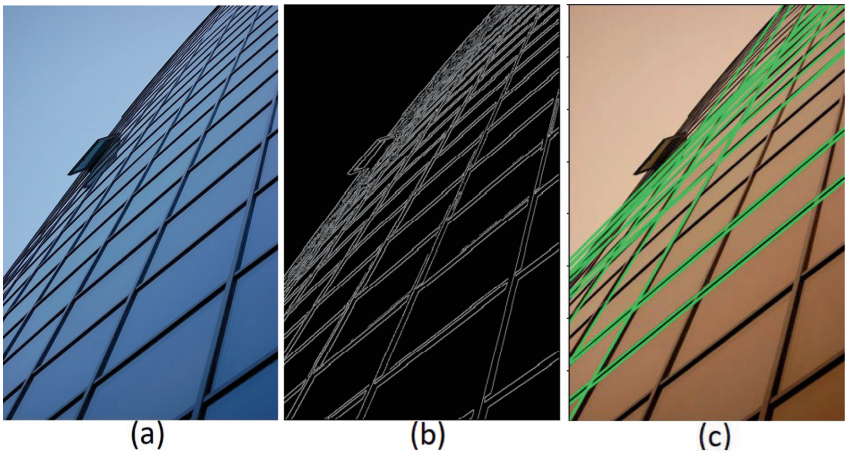


Fig. 5. Straight lines detected on the building with the parameters (threshold = 300, $\alpha = 0.2$, $\gamma = 2$)

4.2 The Case of a Road Image

The experimental results are summarized in the Table 4. The analysis of these results shows that the execution time decreases as the surface of the hexagon increases (we can see these results on Fig. 7(left)). Similarly, if the rate α increases, the execution time decreases (Fig. 7(right)), as well as the number of lines detected (Fig. 7(between)).

4.3 Comparison of Hexagonal Hough Transform and Standard Hough Transform

In order to accentuate the merits inherent in our approach as opposed to the conventional Hough transform, we will undertake a thorough comparative analysis. This involves leveraging a comprehensive evaluation to underscore the distinctive benefits of our method. To facilitate this, we will employ an image of a road (Fig. 6), strategically chosen to serve as a pertinent example for our investigation.

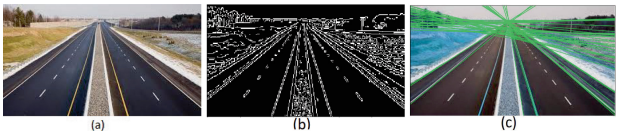


Fig. 6. Straight lines detected on road with the parameters (threshold = 100, $\alpha = 0.1$, $\gamma = 2$)

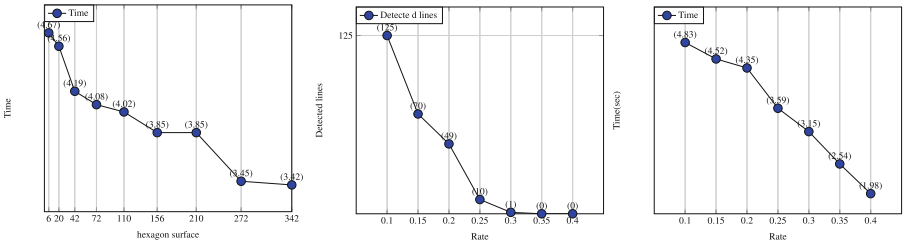


Fig. 7. The road image: (left) the execution time depending on hexagon surface with the parameters (threshold = 100, $\alpha = 0.1$); (between) detected lines depending on the rate with the parameters (threshold = 90, $\gamma = 4$); (right) execution time depending on rate with (threshold = 90, $\gamma = 4$)

The detection is accurate and reasonably fast when the threshold of standard Hough Transform is set high (greater than 100 votes). Table 1 displays the results of using a threshold of 100 votes, where 10 lines were detected in 1.2 s, or 0.12 s per line. On the other hand, the detection deteriorates for lower thresholds, as shown in Fig. 8(b). The latter (Fig. 8(b)) is the result of the standard Hough Transform algorithm with a threshold of 20 votes. With this threshold, 7581 lines were detected in 1.35 s, as illustrated in the first line of the Table 1.

One of the distinct advantages inherent to our proposed method resides in its remarkable flexibility, which can be attributed to the diverse range of input parameters it accommodates. In addition to the pivotal threshold parameter, we have ingeniously incorporated two supplementary parameters, denoted as α and γ. This astute inclusion further empowers users by granting them the ability to finely manipulate and calibrate the behavior of the method to suit specific contexts. The strategic utilization of these parameters presents a unique opportunity to enhance the outcomes delineated in the reference Table 1. By artfully adjusting the values of α and γ, a comprehensive spectrum of optimizations becomes attainable. This dynamic control over the input parameters

not only refines the precision and efficacy of our method but also opens avenues for tailoring it to a multitude of scenarios, thereby bolstering its versatility.

In configuring the parameters to specific values (threshold = 50, $\gamma = 2$, $\alpha = 0.4$), our method detected 10 straight lines in an execution time of 0.84 s. This pivotal achievement is conspicuously showcased in Table 2. In stark comparison, the conventional algorithm took 1.2 s to achieve the same detection result. This substantial improvement is further quantified through the calculation of the acceleration factor denoted as A, where $A = \frac{T_s}{T_h} = \frac{1.2}{0.84} = 1.42$ (T_h represents the execution time of our method's algorithm, while T_s signifies the execution time of the standard algorithm). The fact that this acceleration factor surpasses 1 underscores a clear and tangible enhancement in the swiftness of straight-line detection. Notably, this accelerated performance isn't the sole advantage. The quality of detection is equally exceptional, as evidenced by the resultant Fig. 8(a). The lines demarcating the road's edges are impeccably detected and delineated. This visual validation harmoniously complements the quantitative results, corroborating the robustness and efficacy of our method. When we further lower the threshold to 20 votes, $\gamma = 3$ and $\alpha = 0.35$, 44 lines are detected in 0.53 s as illustrated in the Table 2.

Table 1. Results with Standard Hough Transform applied to a road image(time1 = time of all detected lines; time2 = single line detection time)

Threshold	number of detected lines	time1(sec)	time2(sec)
20	7581	1.35	0.00017
100	10	1.2	0.12

Table 2. Results with Hexagonal Hough Transform applied to a road image(time1 = time of all detected lines; time2 = single line detection time)

γ	$S_H(px^2)$	α	Threshold	number of detected lines	time1(sec)	time2(sec)
2	6	0.4	50	10	0.84	0.084
3	20	0.35	20	44	0.53	0.012

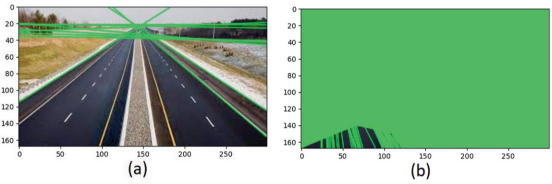


Fig. 8. (a) With Hexagonal Hough Transform (Threshold = 50, $\gamma = 2$, $\alpha = 0.4$); (b) With Standard Hough Transform (Threshold = 20)

The judicious selection of parameters ushered in a significant revolution in the detection process. The execution time was notably reduced, underscored by the acceleration factor, and the precision of detection, as exemplified in the visual output, stands as a testament to the prowess of our method in enhancing both speed and accuracy in line detection.

5 Conclusion and Perspectives

The method of straight-line recognition in an image by the Hough transform is simple and insensitive to noise. It is applicable to almost any type of image and allows you to vary many parameters.

In this paper, analytical straight-line recognition was treated using the Hough transform method on a virtual hexagonal mesh. We have seen that the hexagonal grid has several advantages in the field of image processing. Algorithms based on the standard Hough transform have been proposed to compute the duals of the selected hexagons in order to update the accumulator.

At the end of the analysis of the results of the simulations, there emerges a possibility of reducing the computation time.

Future works will center on enhancing these algorithms by concentrating on execution time refinement through the implementation of parallel programming. Also, we will study the possibility of extending this approach to other types of shapes or patterns beyond straight lines. Additionally, attention will be directed towards integrating Region Of Interest (ROI) practices within image processing to streamline performance. Furthermore, the introduction of artificial intelligence algorithms will be explored as a means to advance these methodologies.

6 Appendix

Table 3. (left) Results with Algorithm 4 applied to a building image, with the parameters (threshold = 200 and $\alpha = 0.3$); (right) Results with Algorithm 4 applied to a building image, with the parameters (threshold = 200 and $\gamma = 4$)

γ	$S_H(px^2)$	number of detected lines	time(sec)
2	6	135	16.32
3	20	27	10.36
4	42	46	9.56
5	72	52	8.94
6	110	34	7.78
7	156	44	6.83
8	210	27	6.24
9	272	38	6.13
10	342	17	6.10

α	number of detected lines	time(sec)
0.10	492	19.57
0.15	441	19.03
0.20	278	15.41
0.25	197	12.36
0.30	46	9.26
0.35	0	5.72
0.40	0	4.42

Table 4. (left) Results with Algorithm 4 applied to a road image, with the parameters (threshold = 100 and $\alpha = 0.1$); (right) Results with Algorithm 4 applied to a road image, with the parameters (threshold = 90 and $\gamma = 4$)

γ	$S_H(px^2)$	number of detected lines	time(sec)
2	6	74	4.67
3	20	74	4.56
4	42	49	4.19
5	72	29	4.08
6	110	29	4.02
7	156	28	3.85
8	210	37	3.85
9	272	15	3.45
10	342	16	3.42

α	number of detected lines	time(sec)
0.10	125	4.83
0.15	70	4.52
0.20	49	4.35
0.25	10	3.59
0.30	1	3.15
0.35	0	2.54
0.40	0	1.98

```
Function count(img : image, A, B, C, D, E, F, G, H : tables of two integers) : real
    Variables: k, l : integers ;
    Begin
        k ← 0; l ← 0;
        MAT1 ← Rely(A,F); MAT2 ← Rely(F,E); MAT3 ← Rely(E,D);
        for A[1] ≤ y ≤ D[1] do
            if A[1] ≤ y ≤ G[1] then
                NP ← |y − A[1]|
                for A[0] − NP ≤ x ≤ A[0] + NP do
                    if (x,y) is an element of MAT1 then
                        do nothing;
                    else
                        if 0 ≤ x ≤ Nl and 0 ≤ y ≤ Nc then
                            if img[x,y] ≠ 0 then
                                k ← k + 1;
                            else
                                l ← l + 1 ;
                            end if
                        end if
                    end if
                end for
            end if
            if G[1] ≤ y ≤ H[1] then
                for B[0] ≤ x ≤ F[0] do
                    if (x,y) is an element of MAT2 then
                        do nothing;
                    else
                        if 0 ≤ x ≤ Nl and 0 ≤ y ≤ Nc then
                            if img[x,y] ≠ 0 then
                                k ← k + 1;
                            else
                                l ← l + 1 ;
                            end if
                        end if
                    end if
                end for
            end if
            if H[1] ≤ y ≤ D[1] then
                MQ ← |y − D[1]|
                for A[0] − MQ ≤ x ≤ A[0] + MQ do
                    if (x,y) is an element of MAT3 then
                        do nothing;
                    else
                        if 0 ≤ x ≤ Nl and 0 ≤ y ≤ Nc then
                            if img[x,y] ≠ 0 then
                                k ← k + 1;
                            else
                                l ← l + 1 ;
                            end if
                        end if
                    end if
                end for
            end if
        end for
        Return: k/(k+l) ;
    end
end
```

Algorithm 4: Computing the Rate for lit pixels

References

1. Reveilles, J.P.: Structures des droites discrètes. In Journées mathématique et informatique. Marseille-Luminy (1989)
2. Mejdani, S.E., Egli, R., Dubeau, F.: Champ de hauteurs de la transformée de Hough standard. Laboratoire MOIVRE, Département d'informatique, Faculté des sciences, Université de Sherbrooke, 2500, boul. de l'Université, J1K2R1, Sherbrooke (Québec), Canada (2005)
3. Hough, P.V.C.: Method and means for recognizing complex patterns. United States Pattent 3069654:47-64 (1962)
4. Duda, R.O., Hart, P.E.: Use of the hough transform to detect lines and curves in pictures. Commun. ACM **15**(1), 11–5 (1972)
5. Davies, E.R.: Finding ellipses using the generalized Hough transform. Pattern Recogn. Lett. **9**(2), 87–96 (1989). https://www.sciencedirect.com/science/article/pii/016786558990041X
6. Coelho, M., Sugimoto, D., Melo, G., Curtis, V., Bezerra, J.: A MapReduce based approach for circle detection. In: Proceedings of the 14th International Conference on Software Technologies - ICSOFT, pp. 454-459. INSTICC. SciTePress (2019)
7. Sere, A., Sie, O., Andres, E.: Extended standard hough transform for analytical line recognition. Int. J. Adv. Comput. Sci. Appl. (IJACSA) **4**(3), 256–266 (2013)
8. Geman, D., Jedynak, B.: An active testing model for tracking roads in satellite images. IEEE Trans. Pattern Anal. Mach. Intell. **18**(8), 1–14 (1996)
9. Benkouider, F., Hamami, L., Abdellaoui, A., Salmon, M.: Extraction de routes par classification superviée et par réseaux de neurones artificiels á partir d'image spot : cas d'une ville oasienne (ALGÉRIE). Teledetection, Editions des Archives Contemporaines/Editions scientifiques GB/Gordon and Breach Scientific (2015). https://halshs.archives-ouvertes.fr/halshs-01133603
10. Sere, A., Traore, C., Traore, Y., Sie, O.: Towards traffic saturation detection based on the hough transform method. In: Arai, K., Kapoor, S., Bhatia, R. (eds.) FTC 2020. AISC, vol. 1289, pp. 263–270. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-63089-8_16
11. Poncelet, N., Cornet, Y.: Transformée de Hough et détection de linéaments sur images satéllitaires et modèles numériques sur terrain. Unité de Géomatique, Département de Géographie, Université de Liège, Liège, Belgique. **54**, 145–56 (2010)
12. Hoppenot, P., Colle, E., Barat, C.: Off line localisation of a mobile robot using ultrasonic measures. Robotica **18**(8), 315–23 (2000)
13. Traore, C.A.D.G.: Application de la transformée de Hough rectangulaire à la détection de droites discrètes. Université Nazi BONI (2019)
14. Ouedraogo, M., Sere, A., Some, B.M.J., Traore, C.A.G.D.: Straight-line recognition using a triangular grid. In: Arai, K. (eds.) Advances in Information and Communication FICC 2022 Lecture Notes in Networks and Systems, vol. 438. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-98012-2_45
15. Sere, A., Ouedraogo, M., Atiampo, A.L.: Parallel hough transform based on object dual and pymp library. Int. J. Adv. Comput. Sci. Appl. **13**(10) (2022). https://doi.org/10.14569/IJACSA.2022.0131084
16. Sere, A., Colazzo, D., Sie, O.: A hough transform based on a map-reduce algorithm. Int. J. Eng. Res. Appl. **6**(8), 7–15 (2016)
17. Curcio, C.A., Pohakr Sloan Jr, K.R.: Distribution of cones in human and monkey retina: individual variability and radial asymmetry (1987)
18. Asharindavida, F., Hundewale, N., Aljahdali, S.H.: Study on Hexagonal Grid in Image Processing (2012)
19. Middleton, L., Sivaswamy, J.: Edge detection in a hexagonal-image processing framework. Image Vision Comput. **19**(14), 1071–1081 (2001). https://www.sciencedirect.com/science/article/pii/S0262885601000671

20. Sahr, K.: Hexagonal discrete global grid systems for geospatial computing. Arch. Photogram. **22**, 363–376 (2011)
21. Varghese, P., Saroja, G.: Edge detection image operators in hexagonal pixel framework. Int. J. Adv. Res. Eng. Technol. **12**, 242–255 (2021)
22. Cha, J., Cofer, R.H., Kozaitis, S.P.: Extended Hough transform for linear feature detection. Pattern Recogn. **39**(5), 1034–1043 (2006)
23. Galambosy, C., Matas, J., Kittler, J.: Progressive probabilistic Hough transform Progressive probabilistic Hough transform for line detection. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1 (1999)
24. Ballard, D.: Hierarchical generalized Hough transform and line segment based generalized Hough transforms. Pattern Recogn. **15**, 277–285 (1982)
25. Kiryati, N., Eldar, Y., Bruckstein, A.M.: A probabilistic hough transform. Pattern Recogn. **24**(4), 303–316 (1991)
26. Luo, D., He, X., Teng, Q., Tao, Q.: Triplet circular Hough transform for circle detection. J. Electron. (China) **19**(4), 356–362 (2002)
27. Chia, A., Leung, M., Eng, H., Rahardja, S.: Ellipse detection with hough transform in one dimensional parametric space. In: IEEE International Conference on ICIP, vol. 5, pp. 333–336 (2007)
28. Lu, W., Tan, J.: Detection of incomplete ellipse in images with strong noise by iterative randomized Hough transform (IRHT). Pattern Recogn. **41**(4), 1268–1279 (2008)
29. Huang, C.L.: Elliptical feature extraction via an improved hough transform. Pattern Recogn. Lett. **10**(2), 93–100 (1989)
30. Daul, C., Graebling, P., Hirsch, E.: From the hough transform to a new approach for the detection and approximation of elliptical arcs. In: Computer Vision and Image Understanding, vol. 72, no. 3, pp. 215–236 (1998)
31. Kalviainen, H., Hirvonen, P., Xu, L., Oja, E.: Probabilistic and non-probabilistic Hough transforms: overview and comparisons. In: Image and Vision Computing, vol. 13, no. 4, pp. 239–252 (1995)
32. Khoshelham, K.: Extending generalized hough transform to detect 3D objects in laser range data. In: ISPRS Workshop on Laser Scanning 2007 and SilviLaser 2007, Espoo, finland, pp. 12–14 (2007)
33. Lopez-Krahe, J., Alamo-Cantarero, T., Davila-Gonzalez, E.: Discrete hough transform applied to small size pattern recognition. éditeur Télécom (1994)
34. Rhody, H., Carlson, C.F.: Hough Circle Transform, Center for Imaging Science, Rochester Institute of Technology (2005)
35. Svalbe, I.D.: Natural representations for straight lines and the Hough transform on discrete arrays. IEEE Trans. Pattern Anal. Mach. Intell. **11**(9), 941–950 (1989)
36. Maji, S., Malik, J.: Object detection using a max-margin Hough transform. In: CVPR 2009, pp. 1038–1045 (2009)
37. Pousset, P., Lopez-kraze, J., Cofer, R.H.: Transformée de hough discrète et bornée, application à la détection de droites parallèles et du réseau routier. Colloque TIPI, 5-N°4, 1988. éditeur Gretsi, Saint Martin d'Hères, France (1988)
38. Ballard, D.H.: Generalizing the Hough transform to detect arbitrary shapes. Pattern Recogn. **13**(2), 111–122 (1981)
39. Vijayarajeswari, R., Parthasarathy, P., Vivekanandan, S., Alavudeen Basha, A.: Classification of mammogram for early detection of breast cancer using SVM classifier and Hough transform. Measurement **146**, 800–805 (2019). ISSN 0263-2241. https://doi.org/10.1016/j.measurement.2019.05.083. https://www.sciencedirect.com/science/article/pii/S0263224119305275

40. Sere, A., Ouedraogo, F.T., Zerbo, B.: An improvement of the standard hough transform method based on geometric shapes. In: Arai, K., Kapoor, S., Bhatia, R. (eds.) FICC 2018. AISC, vol. 887, pp. 369–384. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-03405-4_25
41. Bailey, D., Chang, Y., Le Moan, S.: Analysing arbitrary curves from the line hough transform. J. Imaging **6**, 26 (2020). https://doi.org/10.3390/jimaging6040026
42. Lin, G., Tang, Y., Zou, X., et al.: Fruit detection in natural environment using partial shape matching and probabilistic Hough transform. Precis. Agric. **21**, 160–177 (2020). https://doi.org/10.1007/s11119-019-09662-w
43. Liao, B., Li, J., Ju, Z., Ouyang, G.: Hand gesture recognition with generalized hough transform and DC-CNN using realsense. In: 2018 Eighth International Conference on Information Science and Technology (ICIST), Cordoba, Granada, and Seville, Spain, p. 84–90 (2018). https://doi.org/10.1109/ICIST.2018.8426125.
44. Zhao, K., Han, Q., Zhang, C.-B., Xu, J., Cheng, M.-M.: Deep hough transform for semantic line detection. IEEE Trans. Pattern Anal. Mach. Intell. **44**(9), 4793–4806 (2022). https://doi.org/10.1109/TPAMI.2021.3077129
45. Torrente, M.L., Biasotti, S., Falcidieno, B.: Recognition of feature curves on 3D shapes using an algebraic approach to Hough transforms. Pattern Recogn. **73**, 111–130 (2018). ISSN 0031–3203. https://doi.org/10.1016/j.patcog.2017.08.008. https://www.sciencedirect.com/science/article/pii/S0031320317303096
46. Zhang, C., Wang, F., Zou, Y., Dimyadi, J., Guo, B.H.W., Hou, L.: Automated UAV image-to-BIM registration for building façade inspection using improved generalised Hough transform, Autom. Construct. **153**, 104957 (2023). ISSN 0926–5805. https://doi.org/10.1016/j.autcon.2023.104957. https://www.sciencedirect.com/science/article/pii/S0926580523002170
47. Conti, C., Romani, L., Schenone, D.: Semi-automatic spline fitting of planar curvilinear profiles in digital images using the Hough transform. Pattern Recogn. **74**, 64–76 (2018). ISSN 0031–3203. https://doi.org/10.1016/j.patcog.2017.09.017. https://www.sciencedirect.com/science/article/pii/S0031320317303692
48. Soares, G.A., Abdala, D.D., Escarpinati, M.C.: Plantation rows identification by means of image tiling and hough transform. In: VISIGRAPP, vol. 4. VISAPP (2018)
49. Widyaningrum, E., Gorte, B., Lindenbergh, R.: Automatic building outline extraction from ALS point clouds by ordered points aided hough transform. Remote Sens. **11**(14), 1727 (2019). https://doi.org/10.3390/rs11141727
50. Liang, Q., et al.: Angle aided circle detection based on randomized Hough transform and its application in welding spots detection. Math. Biosci. Eng. **16**(3), 1244–1257 (2019)
51. Marzougui, M., Alasiry, A., Kortli, Y., Baili, J.: A lane tracking method based on progressive probabilistic hough transform. IEEE Access **8**, 84893–84905 (2020). https://doi.org/10.1109/ACCESS.2020.2991930
52. Winterhalter, W., Fleckenstein, F.V., Dornhege, C., Burgard, W.: Crop row detection on tiny plants with the pattern hough transform. IEEE Rob. Autom. Lett. **3**(4), 3394–3401 (2018). https://doi.org/10.1109/LRA.2018.2852841
53. Romanengo, C., Falcidieno, B., Biasotti, S.: Hough transform based recognition of space curves. J. Comput. Appl. Math. **415**, 114504 (2022). ISSN 0377-0427. https://doi.org/10.1016/j.cam.2022.114504. https://www.sciencedirect.com/science/article/pii/S0377042722002448
54. Yang, W., Li, Y., Hu, T., Fuchikami, R., Ikenaga, T.: Relative vectors clustering and temporal constraint based generalized Hough transform for high frame rate and ultra-low delay arbitrary shape detection. In: Proceedings of SPIE 12590, Third International Conference on Computer Vision and Information Technology (CVIT 2022), p. 1259002 (2023). https://doi.org/10.1117/12.2670011

55. Gabrielli, A., Alfonsi, F., Del Corso, F.: Simulated hough transform model optimized for straight-line recognition using frontier FPGA devices. Electronics **11**(4), 517 (2022). https://doi.org/10.3390/electronics11040517
56. Yang, C., Collins, J.: Improvement of honey bee tracking on 2D video with hough transform and kalman filter. J. Sign. Process. Syst. **90**, 1639–1650 (2018). https://doi.org/10.1007/s11265-017-1307-x
57. Freeman, A., Shi, W., Hwang, B.: Enhancing surveillance camera FOV quality via semantic line detection and classification with deep hough transform. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) Workshops, January 2024, pp. 374–380 (2024)