https://dx.doi.org/10.4314/dujopas.v9i1a.5
ISSN (Print): 2476-8316
ISSN (Online): 2635-3490
Dutse Journal of Pure and Applied Sciences (DUJOPAS), *Vol. 9 No. 1a March 2023*

Accelerated Histogram of Oriented Gradients for Human Detection

***Siyudi Shafi'I Umar[1], Zaharaddeen S. Iro[2], Abubakar Y. Zandam[3], Saifulllahi Sadi Shitu**

[1] Computer and Microelectronics Systems Department,
University Teknologi Malaysia.

[2] Computer Science Department,
Federal University Dutse.

[3] Science Education Department,
Jigawa State College of Education
Gumel.

[4] Cyber Security Department,
Nigerian Defence Academy,
Kaduna.

Email: siyudi69@gmail.com

Abstract

Histogram of Oriented Gradients (HOG) is an object detection algorithm used to detect people from an image. It involves features extraction called 'HOG descriptor' which are used to identify a person in the image. Several operations are involved in the feature extraction process. Hence performing numerous computations in order to obtain HOG descriptors takes some considerable amount of time. This slow computation speed limits HOG's application in real-time systems. This paper investigates HOG with a view to improve its speed, modify the feature computation process to develop a faster version of HOG and finally evaluate against existing HOG. The technique of asymptotic notation in particular Big-O notation was applied to each stage of HOG and the complexity for the binning stage was modified. This results in a HOG version with a reduced complexity from n^4 to n^2 thereby having an improved speed as compared to the original HOG.

Keywords: Computer Vision, Object detection, Human Detection, Histogram of Oriented Gradients, Big-O Notation.

INTRODUCTION

Computer vision is a broad field that finds its application in many aspects of our daily lives. It involves enhancement, classification, recognition and detection operations on images and video files we take from our daily activities. It seeks to enable computer system automatically to see, identify and understand the visual world, simulating the same way that human vision does (Feng et al., 2019). Object detection is a widely explored area in computer vision as it has applications in many fields like human computer interaction, smart vehicles, automated manufacturing processes and surveillance systems. It involves detection of a particular class of object from a group of different or similar objects in an image or video. For example,

**Author for Correspondence*

detection of a person from an image containing people, cars, buildings and animals altogether (Zhao et al., 2019).

Human detection is the technology that people usually use to detect objects in images or videos. It is also a crucial step in the video-based surveillance systems. The aim is to identify and monitor humans for security purposes in the crowded environment such as airports, bus terminals or train stations (Hossai et al., 2017). Human detection problems are quite complex due to the several variations in human pose, dress texture and colour. Several approaches were proposed to efficiently tackle these issues among which is the Histogram of Oriented Gradients (HOG) by Navneet Dalal in 2005 (Dalal and Triggs, 2005).

HOG is an object detection algorithm used to detect people and objects from an image. It can extract characteristics about an objects look and shape from distribution of local gradients, it is also capable of defining a distinct texture and shape (Salau and Jain, 2019). HOG-based algorithms are still favourable in many applications due to their balanced trade-off between accuracy and complexity (Ayalew et al., 2022). However, due to the complex procedure involved in HOG feature computation, it takes a considerable amount of time to generate an output vector and hence cannot meet the requirement for real-time applications.

HOG proves that the distribution of gradient or edge directions can be used to describe the local appearances of objects and shapes within an image (Dalal and Triggs, 2005). It involves counting the occurrences of local gradient orientations of objects in xan image. Object features are extracted and are called HOG descriptors. A descriptor computation is achieved by dividing the image into small connected portions called "cells". Histogram of gradient orientations in each cell is computed and the combination of these histograms is what constitutes the descriptor. Group of cells are combined to form a "block" which is then normalized to make the image robust against illumination variations. A typical HOG descriptor computation process is given in Fig.1 below:

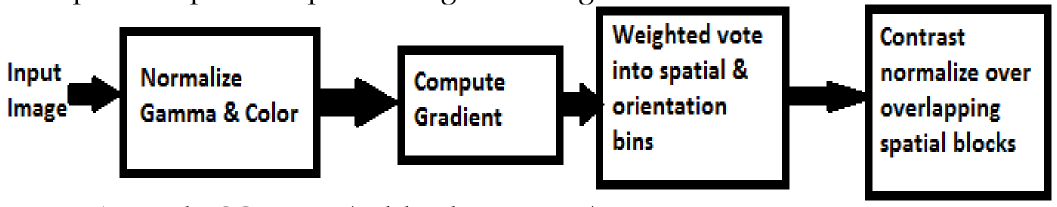
Input Image
Normalize Gamma & Color
Compute Gradient
Weighted vote into spatial & orientation bins
Contrast normalize over overlapping spatial blocks

Figure 1: A typical HOG process (Dalal and Triggs, 2005)

Gamma / colour Normalization

The input image is optionally pre-processed by normalizing it so as to reduce the impact of illumination factor. Two variants of gamma normalization can be applied that is, square root or log compression. The normalization will eliminate noises from capturing device and environment. It is done for each colour channel of the image. An alternative way is to convert the image to grey scale and hence no need to further normalize it.

Gradient Computation

Gradient calculation is basically the first major task in HOG computation. An optional Gaussian smoothing can be applied prior to the gradient computation. Several derivative masks like 1-D point mask, 3 × 3 Sobel mask or 2 × 2 diagonal ones can be used. The performance is however affected by the type of the mask used.
The 1D-centered derivative mask gives the best response and hence is the most widely adopted for the gradient computation.

- 1-D derivative mask

$$D_x = [-1\ 0\ 1]\ and\ D_y = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \qquad (1)$$

The gradient magnitude is computed as the square root as of the quadratic sum of each gradient component, as shown in the equation below

$$M(m,n) = \sqrt{G_x^2\ (m,n) + G_y^2\ (m,n)} \qquad (2)$$

Once the gradient components are obtained, the gradient angle is obtained using the equation below:

$$\emptyset(m,n) = arctan\left(\frac{Gy(m,n)}{Gx(m,n)}\right) \qquad (3)$$

Spatial / Orientation Binning

This task involves dividing the image into small portions called "cells". A histogram of orientation of gradient values within each is plotted where each pixel in the cell cast a weighted vote in the histogram. Histogram channels are created for the orientations usually evenly spread from 0 to 180 degrees (unsigned) or from 0 to 360 degrees (signed) gradient (Wu et al., 2015), thereby resulting in a total of 9 or 18 channels respectively. As highlighted by Dalal (Dalal and Triggs, 2005), the 9 histogram channels configuration outperforms its counterpart. This is due wide range in human clothing and object backgrounds, hence the sign of the angle is usually irrelevant.

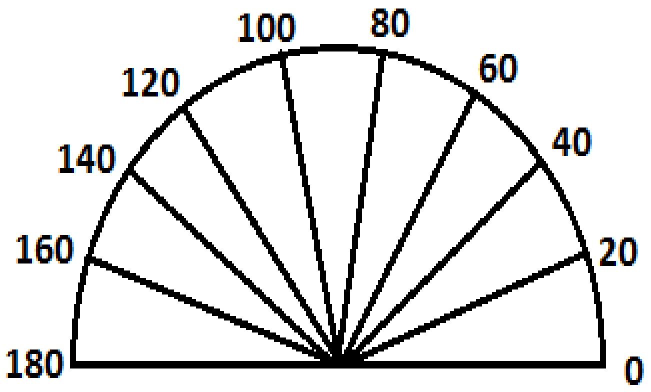

Figure 2: 9 orientations bins configuration (Dalal and Triggs, 2005)

The vote weighting can be either the gradient magnitude, its square or even square root. The gradient magnitude is widely used because it provides the best results. For a typical bin with range say *k,* the equation to compute the *kth* bin on the histogram is given by (Dalal and Triggs, 2005)

$$h_k = \sum_{m,n} = \ M(m,n)\ 1\ [\varnothing(m,n) = k] \qquad (4)$$

Descriptor Blocks

Due to effects of variations in illumination and contrast, the gradients need to be strengthened so that they become robust to these adverse effects. This is achieved by combining a group cells to form a "block" and the blocks are locally normalized. Over all the block regions, the vector component of the normalized cell histograms gives to what is called the "HOG descriptor". Each cell normally contributes to more than one block due to the block-overlapping with neighbouring ones (Yuan et al., 2015). The two main variants of the descriptor blocks are the Rectangular-HOG (R-HOG) and the circular-HOG (C-HOG). Fig. 3. below shows the various cells and the grouping of cells to form a block.

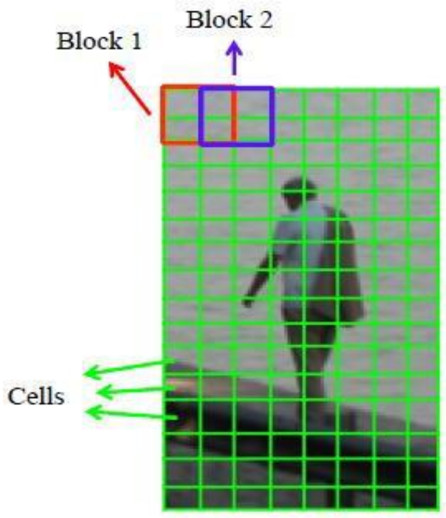

Figure 3: Descriptor block and cells formation (Dalal and Triggs, 2005)

Block Normalization

Different methods can be used for block normalization as a counter measure to effects of illumination and contrast variations (Zhao et al., 2015). The two major ones are L1- norm and L2-norm.

All the above stages have to be carried out in order to compute HOG descriptors of any input image which are then passed to a classifier for grouping of datasets and detection. This prompts many researchers to work on improving the detection stage by implementing HOG with a combination of other pre-processing or post processing techniques. For example, Shihong Yao *et al.* surveyed several combinations of HOG with other algorithms like Scale Invariant Feature Transform (SIFT), Speeded-up Robust Feature (SURF), Local Binary Pattern (LBP), Multi-scale Orientation (MSO) and Local Self Similarities (LSS). The HOG-LSS combination was found to have the strongest description ability and relatively faster than the other combinations**.** LSS is used locally to extract self-similarity feature of edges, colours, complex textures and patterns in a unified format. Its four main parameters are: image size, window radius, angle interval and interval radius of image patches. HOG-LSS gives a combined descriptor that contains both local and global information about an image. It also has the ability to capture rough and contour information (Yao et al., 2015).

Surasak et. al implemented a human detection system using Python and OpenCV. HOG was used as the main detection algorithm with Support Vector Machine (SVM) as classifier. They were able to achieve a detection accuracy of 81.23% with a standard deviation (SD) of 10.95%. The system developed has the whole execution time about 5-10 minutes per cycle - which is quite long. To solve this problem, they applied four threads parallel execution features using Python Multiprocessing Library to reduce the processing time by 45.96% (Surasak et al., 2018). A study by Zhou et. al, the origin of the image was explored with a view to handle the redundancy associated with the input image processing pipeline. Cumulative errors are added to source image by image sensors in the process of producing an optimal quality image. Using the minimal image processing pipeline which consist of denoising, demosaicing and gamma compression, the study was able to experiment that by taking advantage of the inter-channel correlation of natural images, the HOG features can be directly extracted from the Bayer pattern images, where proper gamma compression may be needed. An evaluation was done between images generated using minimal image processing pipeline and those generated from raw Bayer pattern images after applying gamma compression. The result showed that for HOG-based pedestrian detection systems, the complete image processing pipeline except the gamma compression can be skipped. Hence, the HOG features based on Bayer pattern images can be used in pedestrian detection algorithms with little performance degradation (Zhou et al., 2020).

Categorically, the efforts done by researchers to improve HOG can be broadly classified into Implementation approach and Algorithm approach. The former involves running HOG on a faster device, different platform or in a parallel mode. For example executing HOG on a very fast Central Processing Unit (CPU), in parallel on a Graphics Processing Unit (GPU) and using Field Programmable Gate Array (FPGA). The latter involves modifying the HOG algorithm itself. For example, combining HOG with some other algorithm and improving image acquisition and enhancement procedure before computing its HOG feature. This works aims to explore HOG as human detection algorithm with a view to overcome some of its limitations in particular – computation speed. The algorithm approach will be adopted due its low cost, accuracy and better resource utilization. The time complexity of each stage of HOG will be studied with a view to identify the slowest or compute intensive stage(s) using Asymptotic notation analysis. The identified stage(s) will be improved at algorithm level for faster completion keeping in mind the overall accuracy of the process. In doing so, the original HOG will still be maintained without adding other algorithms which may have their own limitations and an improved version will be obtained.

METHODOLOGY

Based on the scope of this work, HOG speed will be improved at the algorithm level. Focus will be made on the HOG descriptor computation stage and the appropriate techniques will be applied to it in order to get the desired result. Image datasets will be collected from samples of INRIA still image data sets. The coding and simulation will be carried out using MATLAB software tool on a PC having a dual core Intel® Pentium® Processor clocked at 2.1GHz with 4GB RAM running Windows 7 Ultimate 64bit OS.

The complete methodology flow is shown in Fig. 4 below:

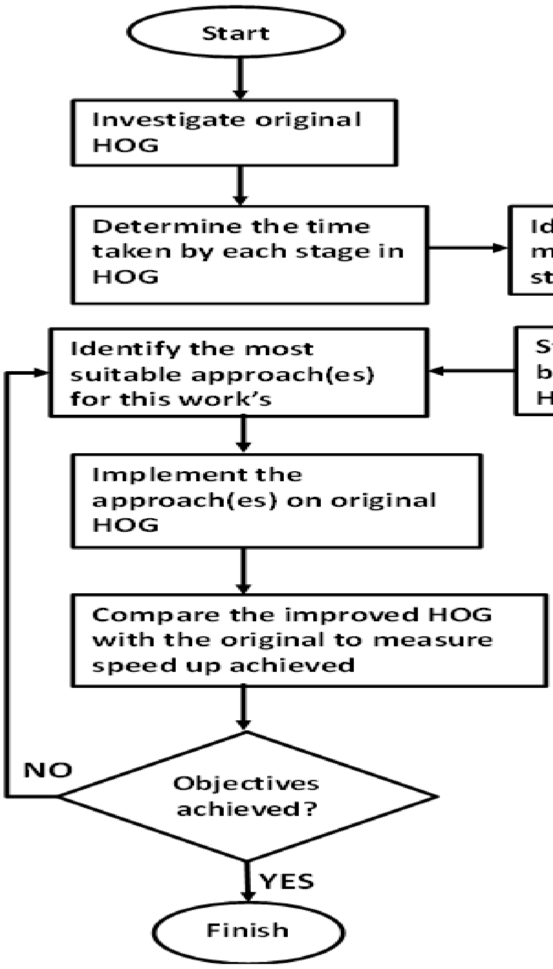
Start
Investigate original HOG
Determine the time taken by each stage in HOG
Identify the slowest or most time consuming stage
Study approaches taken by others to improve HOG
Identify the most suitable approach(es) for this work's
Implement the approach(es) on original HOG
Compare the improved HOG with the original to measure speed up achieved
Objectives achieved?
NO
YES
Finish

Figure 4. Complete methodology flow chart.

HOG Time Complexity: Asymptotic notations is a technique in the study of algorithms that focuses on characterization according to their efficiency. The key area here is the growth rate of the algorithm with respect to input size (Bensoussan et al., 2011, Ye et al., 2021). Emphasis is made on the order of the increase in running time of an algorithm. This is because the growth rate provides a way of classifying and comparing algorithms unlike the measure of their exact running time. Once the order of growth of functions is known, the basis for classification and comparison is obtained as well. Any typical computer algorithm involves a lot of steps or procedures; the most common ones are mathematical operations. These operations are being executed in specified manner to achieve a task which is the algorithm output. The number and type of operations that are involved in an algorithm is a key factor in determining the efficiency of the algorithm. As a good algorithm is expected to withstand the worst-case scenario with respect input size, asymptotic notation proves to be valuable in algorithm design. Several factors contribute to the running time of an algorithm for example the programming language used, the compiler used and the computer system on which it is run. It is normally carried out for three major cases:

1) Best case scenario also called big Omega - **Ω**
2) Equivalent case scenario also called big theta - **θ**
3) Worst case scenario also called big O - O

The case of emphasis here is the worst case scenario also called as Big-O. It gives an analysis measure of the maximum amount of time taken by an algorithm for a large input size. It gives the upper bound for the growth of the running time of the algorithm and hence called 'upper asymptotic bound'. It is denoted by the upper case alphabet – O. Big-O is used for asymptotic notation for the worst case scenario or ceiling growth rate of a function. The maximum amount of time taken by an algorithm can be obtained using this notation. It bounds the growth of the running time from an upper region for large enough input sizes. It is the most widely used notation as we are more concerned with handling worst case eventualities. The big-Oh notation is used widely to characterize running times and space bounds in terms of some parameter 'n', which varies from problem to problem, for example in our case 'n' is considered to be input image size.

Suppose f(n) and g(n) are functions mapping non negative integers to real numbers. We say that f(n) is O(g(n)) if there is a real constant c > 0 and an integer constant n_o >=1 such that f(n) =< cg(n) for every integer n >= n_o (Goodrich and Tamassia, 2001).
This definition is often referred to as the "big-Oh" notation, for it is sometimes pronounced as "f(n) is big- Oh of g(n)". Alternatively, we can also say "f(n) is order g(n)."

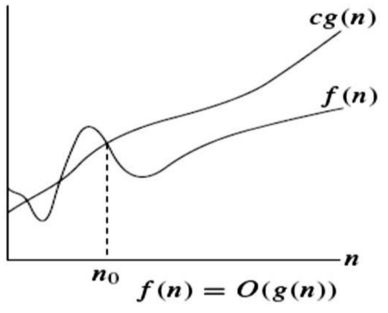

Figure 5 Big-O notation

Asymptotic notation allows us to classify functions according to their growth rate. The growth rate is classified from the slowest to the fastest growing function, which is analogous to from fastest execution time to the slowest execution time. A logarithmic function is the slowest growing function whereas an exponential function is the fastest growing function. These functions are summarized in the table below in ascending order of their growth rate where n is input size and c is a constant:

Table 1: Functions hierarchy in order of complexity

Notation	**Name**
O(1)	Constant
O(log(n))	Logarithmic
O(n)	Linear
O(n^2)	Quadratic
O(n^c)	Polynomial
O(c^n)	Exponential

Asymptotic notation allows us to ignore constants in a complexity equation. This is because as the input size becomes larger, the constant factors only contributes a little to running time overhead. This also applies to lower order terms in the complexity equations as well because at larger inputs, only the highest order term has much influence on the growth rate (Sedgewick and Wayne, 2014, Kalonda and Omekanda, 2020).

The idea behind these is due to the useful fact that:
"When a function say $f(n)$ is formed by a summation of other functions for example $f_1(n)$ and $f_2(n)$ where $f_1(n)$ grows faster than $f_2(n)$, therefore the order of $f(n)$ is determined by $f_1(n)$ since it is the faster growing function" (Wegener, 2005).

The timing complexity of an algorithm is primary factor in determining its efficiency. It normally gives the relationship between size of input data and the computation time required to process it. Its impact is not very noticeable in simple algorithms for example string concatenation, or when the input data size is small for example a 2-by-2 array. Time complexity comes into significance when the algorithm function of concern involves complex procedures like sorting or recursive operations on a very large growing input data size . A less time complexity that is, less computation time algorithm is generally considered to be a more efficient algorithm as compared to one that consumes longer computation time. HOG involves several operations on an image most which requires repeating similar procedure on every pixel in an image (Zhou et al., 2020). It can be seen that as the image size gets larger, these operations also increased, hence resulting in a longer computation time to generate output descriptors.

HOG involves stages and each stage involves a number of operations. These stages were analysed and the complexity equation for each was derived. The complexity analysis is done for the upper bound or worst case scenario using Big - Oh notation. For complete HOG process, the complexity equation is given by the summation of the order of growths of the HOG stages. This is shown in the relationship below:

$$f(HOG) = f(Grad) + f(Bin) + f(Norm) + f(khog) \qquad (5)$$

where *khog* is a constant
Considering the stages individually;
The Gradient computation:

$$\begin{aligned} f(Grad) &= f(\text{grad mag}) + f(\text{grad angles}) + f(\text{kgrad}) \\ &= O(\text{n} \times \text{m}) + O(\text{n} \times \text{m}) + O(\text{kgrad}) \qquad (6) \end{aligned}$$

The Binning Stage:

$$f(Bin) = O\left(\left(\frac{\text{n}}{\text{c}} - 2\right) \times \left(\frac{\text{m}}{\text{c}} - 2\right) \times (\text{b} \times \text{b}) \times (\text{n} - 2) \times (\text{m} - 2)\right) + \text{O}(\text{kbin}) \qquad (7)$$

The Block Normalization Stage:

$$f(Norm) = O[(\text{n} - 2)(\text{m} - 2)] + O(\text{knorm}) \qquad (8)$$

where n is input image height, m is input image width, c is cell division, b is number of orientation bins and *kgrad/kbin/knorm* are constants.

As stated earlier, asymptotic notation allows us to ignore constants in a complexity equation. This is because as the input size becomes larger, the constant factors only contributes a little to running time overhead. This also applies to lower order terms in the complexity equations as well because at larger inputs, only the highest order term has much influence on the growth rate. Hence, complexity equations above can be further simplified by ignoring all constants and dropping all lower order terms. The resulting equations are as below:

$$f(Grad) = O(n^2) \qquad (9)$$

$$f(Bin) = O(n^4) \qquad (10)$$

$$f(Norm) = O(n^2) \qquad (11)$$

Therefore,

$$f(HOG) = O(n^2) + O(n^4) + O(n^2)$$
$$= O(n^4) \qquad (12)$$

Hence the test HOG used is a polynomial function with an upper asymptotic bound of O(n^4). The relationship between the growth rate and input size is of the polynomial order.
It was observed that the most time consuming stage is the binning stage. Its modified complexity equation is as below:

$$f(Bin) = O([({}^{n}/_{c} - 2) \times ({}^{m}/_{c}) - 2 \times (b \times b) \times (c \times c)]) + O(kbin))$$
$$= O\left(\frac{nm}{c^2} - \frac{2(n+m)}{c} + 4\right) \times b^2c^2 + O(kbin)$$
$$= O(b^2nm) + O\big(2cb^4(n+m)\big) + O(kbin) \qquad (13)$$

Where *n* & *m* are input image dimension, *c* is cell dimension, *b* is number of bins and *kbin is* constant.

RESULTS

Asymptotic notations allow us to analyze and develop time complexity equations for algorithms. These give us an insight into the time required by an algorithm in operation. As we always prepare to handle worst case performance conditions, Big O notation provides us a technique of knowing the upper asymptotic bound of the running time of an algorithm (Bensoussan et al., 2011). The HOG complexity equations developed in the previous chapter were modified with a view to obtain a faster HOG version. The modification is done in such a way that HOG parameters are less dependent on the input image size. It involves assuming some parameters to be constants while other are left to vary as the input image size varies.
Summing the complexity of all the HOG stages above to obtain the general complexity equation of the HOG algorithm yields;

$$f(HOG) = f(Grad) + f(Bin) + f(Norm)$$
$$= 20(n \times m) + O(b^2nm) + O\big(2cb^2(n+m)\big) + O(nm - 2(n+m) + 4) + O(kgrad) + O(kbin) + O(knorm) \qquad (14)$$

For large input values, the constants, scalar multiplicands and lower order terms can be ignored. Hence the equation above yields:

$$f(HOG) = O(n \times m) \qquad (15)$$

In a scenario where n=m;

$$f(HOG) = O(n^2) \qquad (16)$$

A reduction in the complexity implies slower growth rate and hence a faster running time. The tuned parameter here is the cell size. It is made to be independent from the input image size. This was tested for various cell sizes in order to find the near optimal cells size for efficiency in both speed and accuracy. Cells sizes of 4 × 4, 6 × 6 and 8 × 8 were tested for varying image sizes. A 16 × 16 cells size gave a very poor timing performance and hence was discarded.

Table 2 below shows the computations time obtained for the different cell sizes.

Table 2: Computation for varying image sizes using different cells sizes.

Image Size (m × n)	Cells size computation time		
	4 × 4	6 × 6	8 × 8
64 × 128	0.393	0.411	0.661
128 × 256	0.847	1.692	2.819
256 × 512	3.540	6.895	11.821
640 × 480	8.393	16.754	27.983
720 × 480	9.399	18.785	31.416
852 × 480	11.113	22.335	37.278
786 × 576	12.218	24.398	41.552
1280 × 720	24.532	49.159	84.129
1920 × 1080	56.700	112.380	189.024

Table 2 above shows the computation time obtained with different input image size and cell sizes. A smaller input image size combined with a cell size of 4 × 4 gives the fastest computation time. The computation time can be seen to increase gradually with increase in cell size but rapidly with increase in input image size. The cells sizes can be increased above 8 × 8 however this normally gives a very poor performance. The performances recorded above were depicted graphically in the Fig. 6 below:

Figure 6 Running time using different cell sizes

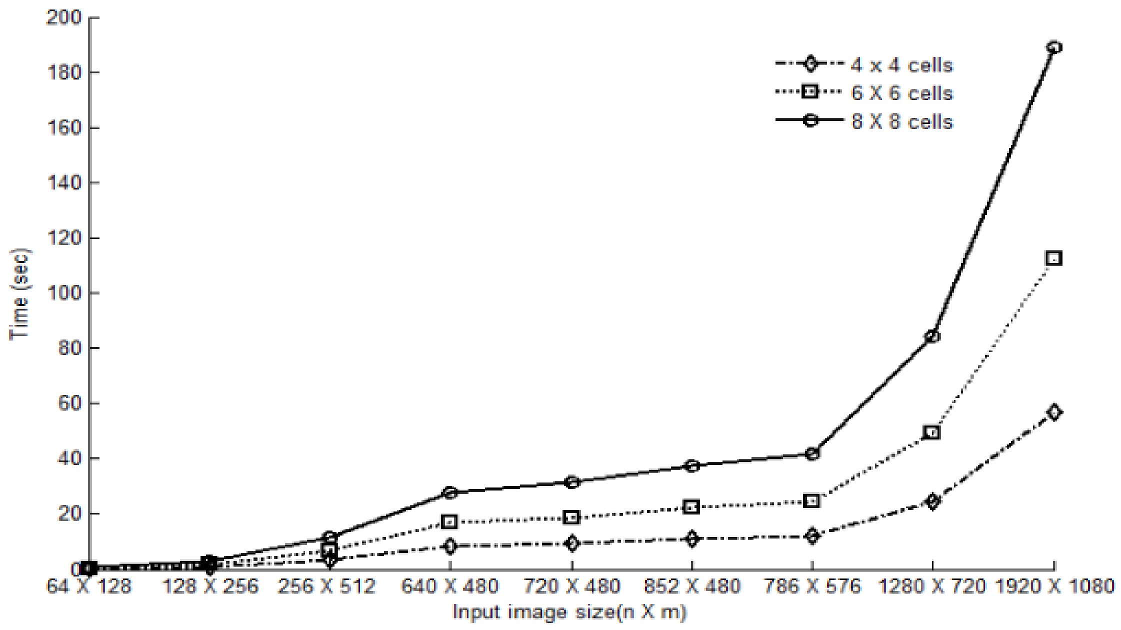

Fig. 6 above depicts the graph of computation time against varying input image size for 3 cell sizes. The smaller cell size of 4 × 4 gives the fastest computation time. The computation time can be seen to increase gradually with increase in cell size (from 4 × 4 to 6 × 6) but rapidly with increase in input image size.

The two HOG variants testHOG and thisHOG were run side by side on the same platform using the 6 × 6 cell size. The computation time obtained on different input image sizes for each is shown below:

Table 3: Computation time between testHOG and thisHOG

Input image Size (n × m)	HOG Computation Time	
	testHOG	thisHOG
64 × 128	0.857	0.411
128 × 256	2.825	1.642
256 × 512	11.700	6.810
640 × 480	28.068	16.190
720 × 480	31.571	18.187
852 × 840	37.703	21.697
786 × 576	42.038	24.126
1280 × 720	85.319	49.122
1920 × 1080	191.631	110.134

The significant decreases in growth can be observed from Table 3 above where the growth rate seems to be reduced by a half in the modified HOG. This is as expected since the time complexity was reduced from n^4 to n^2. This is further depicted Fig. 7 below:

Figure 7: Growth rate for testHOG and thisHOG

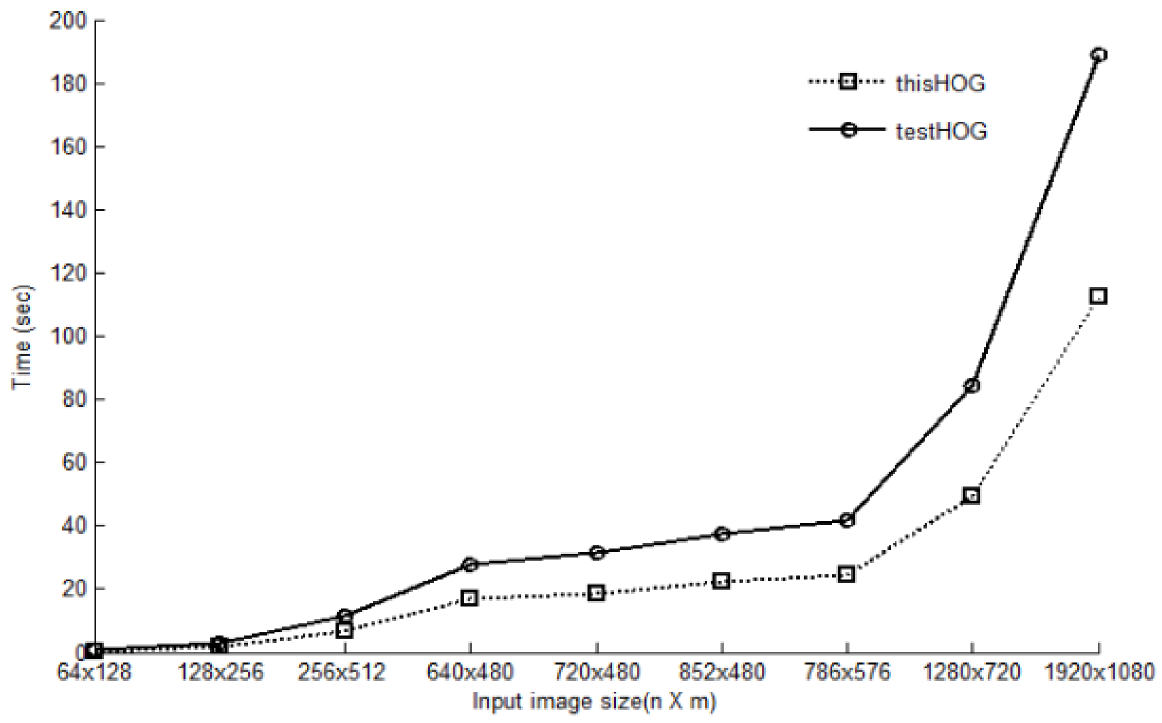

Fig. 7 above shows the computation time taken by each HOG version using a 6 × 6 cell size for various input image size. It can be seen that the time increase steadily at smaller size input images but increases rapidly for larger size input images.

DISCUSSION

The 3 main stages of HOG algorithm i.e Gradient stage, Binning stage and Block Normalization stage (Zhou et al., 2020) were analyzed using Asymptotic notation method. Worst case scenario analysis or Big O notation was applied to the complexity equation of each stage. The gradient and block normalization stage were found to have a complexity of quadratic function (n^2). The binning stage was found to have a polynomial function complexity (n^4). It also happens to be the most compute intensive stage and hence it contributed the most in the long execution time of the algorithm. It was observed that the number of operations carried out in the binning stage largely depend on the size of the input image. Big O notation was applied to this stage and its complexity was reduced from polynomial to quadratic order (from n^4 to n^2).

The modified binning stage was used to form a variant of the HOG algorithm with a quadratic function complexity. This was named 'thisHOG' and was implemented using 3 different cell sizes of 4 × 4, 6 × 6 and 8 × 8. Input images of various sizes were used and their respective computation time was recorded. As can be seen from Table 2, for an input image of size 64 × 128 the time taken by the cell configurations are relatively close 0.393, 0.411 and 0.661 respectively. However, by increasing the input image size to 128 × 256, the computation time taken rapidly increases. For example it quadrupled in the case of 6 × 6 and 8 × 8 cell configuration.The 4x4 cell time doubled and since it has a smaller cell size, it will require more number of cells to carry out a complete feature extraction process depending on the input image size. Also Fig. 6 shows how the computation time rapidly increases with large input images especially for the 8 × 8 cell configuration. This point to us that for larger input images, higher cell configurations take longer duration. This is the reason why cell size of 16 ×16 was dropped and infact any cell size higher than 8 × 8 will not give the desired speed improvement.

The original HOG algorithm (testHOG) and the modified version (thisHOG) were run on the same platform using 3 different cell sizes to evaluate their performance. The key area of focus here is the computation time of each algorithm. From Fig. 6 above, the 4 × 4 cells size was found to be have the slowest growth rate hence the fastest. However, it is more likely to result in too many cells to perform operations on. The 8 × 8 cells sizes is a quite good but was found to have a very fast growth rate as the input becomes larger. For a balance between speed and accuracy, the 6 × 6 cells size will be more suitable as it has a moderate growth rate even when the input size is very large.

The HOG algorithm used for test case in this work was compared with the modified HOG adopting the 6 × 6 cells size variant. This is synonymous to comparing an n^4 complexity algorithm with an n^2 complexity algorithm. The n^2 complexity algorithm (modified HOG) as expected proves to have a slower growth rate and a faster running time with respect to the n^4 complexity algorithm (testHOG).

CONCLUSION

This work focuses on HOG as an algorithm for human detection by investigating the time complexity of the individual stages in the algorithm. The most complex and time consuming stage which is the voting and binning stage was modified using Big-oh asymptotic notation technique. Its time complexity was successfully reduced from polynomial order of n^4 to a quadratic order of n^2. The modified algorithm was tested for varying image sizes using different cells sizes. The 6 x 6 cells size was found to provide a better speed performance as compared to other configurations.

REFERENCES

Ayalew, A. M., Salau, A. O., Abeje, B. T. & Enyew, B. 2022. Detection and classification of COVID-19 disease from X-ray images using convolutional neural networks and histogram of oriented gradients. *Biomedical Signal Processing and Control,* 74**,** 103530.

Bensoussan, A., Lions, J.-L. & Papanicolaou, G. 2011. *Asymptotic analysis for periodic structures*, American Mathematical Soc.

Dalal, N. & Triggs, B. Histograms of oriented gradients for human detection. 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05), 2005. Ieee, 886-893.

Feng, X., Jiang, Y., Yang, X., Du, M. & Li, X. 2019. Computer vision algorithms and hardware implementations: A survey. *Integration,* 69**,** 309-320.

Goodrich, M. T. & Tamassia, R. 2001. *Algorithm design: foundations, analysis, and internet examples*, John Wiley & Sons.

Hossai, M. R. T., Shahjalal, M. A. & Nuri, N. F. Design of an IoT based autonomous vehicle with the aid of computer vision. 2017 International Conference on Electrical, Computer and Communication Engineering (ECCE), 2017. IEEE, 752-756.

Kalonda, P. O. & Omekanda, A. M. Kruskal's Algorithm, Vogel's Approximation and Modified Distribution Methods for the Design of Optimal Electrical Networks in the Democratic Republic of Congo. 2020 IEEE PES/IAS PowerAfrica, 2020. IEEE, 1-5.

Salau, A. O. & Jain, S. Feature extraction: a survey of the types, techniques, applications. 2019 International Conference on Signal Processing and Communication (ICSC), 2019. IEEE, 158-164.

Sedgewick, R. & Wayne, K. 2014. *Algorithms: Part I*, Addison-Wesley Professional.

Surasak, T., Takahiro, I., Cheng, C.-H., Wang, C.-E. & Sheng, P.-Y. Histogram of oriented gradients for human detection in video. 2018 5th International conference on business and industrial research (ICBIR), 2018. IEEE, 172-176.

Wegener, I. 2005. *Complexity theory: exploring the limits of efficient algorithms*, Springer Science & Business Media.

Wu, S., Laganiere, R. & Payeur, P. 2015. Improving pedestrian detection with selective gradient self-similarity feature. *Pattern Recognition,* 48**,** 2364-2376.

Yao, S., Pan, S., Wang, T., Zheng, C., Shen, W. & Chong, Y. 2015. A new pedestrian detection method based on combined HOG and LSS features. *Neurocomputing,* 151**,** 1006-1014.

Ye, C., Cui, J. & Dong, H. 2021. Asymptotic Analysis of Nonlinear Robin-Type Boundary Value Problems with Small Periodic Structure. *Multiscale Modeling & Simulation,* 19**,** 830-845.

Yuan, Y., Lu, X. & Chen, X. 2015. Multi-spectral pedestrian detection. *Signal Processing,* 110**,** 94-100.

Zhao, X., He, Z., Zhang, S. & Liang, D. 2015. Robust pedestrian detection in thermal infrared imagery using a shape distribution histogram feature and modified sparse representation classification. *Pattern Recognition,* 48**,** 1947-1960.

Zhao, Z.-Q., Zheng, P., Xu, S.-T. & Wu, X. 2019. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems,* 30**,** 3212-3232.

Zhou, W., Gao, S., Zhang, L. & Lou, X. 2020. Histogram of oriented gradients feature extraction from raw Bayer pattern images. *IEEE Transactions on Circuits and Systems II: Express Briefs,* 67**,** 946-950.