

# Low Cost and High-Speed Implementation of K-Means Algorithm for Image Analysis

Anuradha. M.G<sup>1\*</sup>, L Basavaraj<sup>2</sup>

<sup>1</sup>Department of Electronics and Communication, JSS Academy of Technical Education, (Affiliated to Visvesvaraya Technological University, Belagavi), Dr. Vishnuvardhan road, Bengaluru-560060, India

<sup>2</sup>Department of Electronics & Communication, ATME College of Engineering, Bannur Rd, Mysuru, Karnataka 570028.

\*Corresponding Author Email: [anuradhamg@jssateb.ac.in](mailto:anuradhamg@jssateb.ac.in)

## ARTICLE INFO

Received: 19 Dec 2024

Revised: 29 Jan 2025

Accepted: 12 Feb 2025

## ABSTRACT

**Introduction:** An image contains various patterns which need to be divided for further analysis. The categorization of patterns can be carried out using unsupervised clustering algorithm of which K-Means is the most popular and efficient algorithm. However, the algorithm takes more time to divide the image into various groups as it is iterative and is computationally intensive.

**Objectives:** Hence introduced a new technique and have devised a new architecture to accelerate the K-Means operation. The technique uses image intensity levels without the spatial information to accelerate the clustering operation. In any image, regardless of its pixel count, the intensity values are always between 0 and 255. Once these intensity values are initially read for an image, subsequent clustering iterations only use the frequency of each intensity level and 256 intensity levels to cluster.

**Methods:** This approach minimizes storage requirements and speeds up the algorithm. Absolute distance is used to compare the distance between the input image pixel and the cluster center to further accelerate the operation. An integer divider is employed to calculate the cluster center. Its architecture is designed to function within a single clock cycle, enhancing the overall speed of the operation

**Results:** Experimental results show efficacy of the proposed architecture in speeding up the operation. When implemented on a Virtex-6 FPGA, the hardware accelerator can process 405 image frames per second for images of size 256 x 256. To adapt the design for a more cost-effective FPGA with fewer logic gates, parallel computations were reduced, allowing implementation on a Spartan 3 FPGA.

**Conclusions:** The modified design processes 106 images per second on the Spartan 3, making it a compelling solution for clustering images with both low cost and high speed.

**Keywords:** Computer vision, Unsupervised clustering, K-means, FPGA, Hardware accelerator.

## 1. INTRODUCTION:

A human being with his inherent ability can analyze any given image. To mimic the behavior of a human brain, machines are being used. A machine can understand the pattern of an image by proper grouping or clustering. Clustering involves in partitioning the vast data into certain number of groups. The data within a group has similar characteristics while the data in different groups are dissimilar. Many clustering algorithms are available in literature to group an image in real time such as K-Means, K-Median, K-Mode, Mean shift algorithm, KD tree, hash table, fuzzy c-Means algorithm, density based algorithms and model based clustering algorithms. Of the available algorithms, K-Means is the algorithm that is explored more due to its inherent simplicity and high efficiency. It is used in variety of multimedia applications like quantizing an image, segmenting video and image streams, recognition of speech and compression of video and audio.

In 1967, Mac Queen proposed the famous K-Means clustering algorithm [1]. The algorithm was used by many researchers from then due to its robustness and simplicity. The algorithm is iterated till it is converged to a local optimum solution by minimizing the squared error between the cluster center and each input data. Due to this comparison, the algorithm requires more computational resource and is slow. Also the algorithm is converged after several iterations. Initially till early 2000's, the algorithm was implemented in software and hence was slower.

To improve the speed of operation, hardware-software co-design technique was implemented [2] which increased the speed 2 times compared to software implementations only. Since then, to accelerate the speed of operation, many researchers used hardware. As a first attempt, Estlic [4] used FPGA to compute the distance and mean computation was carried out using a microprocessor. The author showed that the hardware implementation was carrying the operation 2 times faster than the algorithm built in software that is run in Pentium 3 machine. With the advancement in technology, to accelerate the operation, many hardware architectures were proposed for various applications like background subtraction, image segmentation and color quantization [5]. To improve the speed, many techniques were used. To accelerate the speed, Author [6] read the entire image data for the first time and later used only boundary points in successive iteration. An adaptive K-means algorithm was proposed by Boris Maliatski [7] which could process 15 frames per second. Author Chen [8-12] has built many hardware accelerators using TSMC 90nm technology for K-means algorithm each addressing various issues of the algorithm. Each of the architecture built by the author operates at high speed but the maximum number of image frames that can be operated is limited to less than 20 frames. The author [13] has implemented K-means on Virtex4 FPGA for bio-informatics application. The author shows that the hardware accelerator built on FPGA works 10 times faster than the software counterpart implemented using Matlab which is running in Intel Pentium core duo processor operating at 3GHz speed. Deng [14] uses histogram as an input to accelerate the speed of clustering an image. The author shows that 95-140 images frames of size 640X480 can be clustered in a second using FPGA. Further Farivar[15] explored the use of Nvidia graphics card to perform clustering operation. He has experimentally shown that the clustering operation can process 6 image frames per second.

Even though many accelerators are available in literature, the number of image frames that can be processed by them still remains a challenge as image resolution is also increasing with advent of technology. The computational complexity involved in clustering an image is increasing and new technique has to be adopted to group the image pixel at real time.

The contributions of the article are as follows:

- Efficient architecture is proposed to compute the distance between the image pixel and cluster centers.
- The literature earlier proposed takes image input as multi-dimensional and compares the input each time with the centers. The proposed accelerator considers only the intensity value of an image as input. Each pixel input is not compared with the cluster center. But instead, each pixel input is compared with the possible intensity values which may vary from 0 to 255 in any image and count of the corresponding intensity is incremented. Later only the intensity values are compared with the cluster centers in further iterations reducing the computations to only 256 irrespective of the image size and hence computations can be done quickly compared to the previous hardware architectures available in the literature. Hence the proposed accelerator operates very with less area and power. The proposed method can process 406 image frames in a second and can process an image of size 16 M pixel using the same accelerator.
- The computation of the new cluster center needs division operation which takes multi-clock cycle making the hardware pipeline difficult. A new methodology for division is proposed which operates in one cycle so that the clustering operation can be operated as pipeline to increase the throughput of the clustering operation.

Further the manuscript is organized as follows: The introduction and a brief survey are provided in section 1. Section 2 provides the methodology used to formulate the architecture with each subsection explaining the hardware architecture of each of the block used. Section 3 discusses the result and Section 4 summarizes the research result.

## 2. FORMULATED ALGORITHM AND ARCHITECTURE

The data or an image pixel that needs to be clustered into K groups is represented by 8 bit as illustrated in equation (1)

$$x_i = \{x_{i7}, x_{i6}, x_{i5}, \dots, x_{i1}, x_{i0}\} \quad (1)$$

Let there be 'n' such image pixels that needs to be clustered into 'K' groups. Let the cluster centers of each group are represented as C1, C2...CK. The architecture are capable of dividing the images to maximum of 25 groups and hence K can vary from 1 to 25. Initially, the 'K' cluster centers are chosen randomly from an input image which speeds up the operation. The algorithmic steps used to implement the architecture are given below:

**Step 1:** Read one input data pixel  $x_i$  in a clock cycle.

**Step 2:** Compare the input pixel value  $x_i$  with the intensity values  $m_j$  which has values fixed values '0' to '255' and increment the corresponding register  $N_j$  which hold the number of pixels having the same intensity value. Hence there are 256 registers to hold the corresponding intensity values. Equation (2) gives mathematical notation of the same.

$$N_j = \begin{cases} N_j + 1 & \text{if } (x_i == m_j) \\ N_j & \text{otherwise} \end{cases} \quad (2)$$

The value of j and i vary from 0 to 255 and from 0 to (N-1) respectively.

**Step 3:** Repeat Step1 and Step2 to read and compare all the inputs in N clock cycles.

**Step 4:** Calculate the distance between each intensity value with all the cluster centers using absolute distance calculation as shown in equation (3).

$$d_j = |m_j - c_k| \quad (3)$$

To compare the distance, squared distance computations can also be used as indicated in equation (4)

$$d_j = (m_j - c_k)^2 \quad (4)$$

Step 5: Assign the intensity value  $m_j$  to the nearest cluster centers as described in equation (5)

$$N\left(\frac{x}{m_j}\right) = C_k \text{ if } x = \operatorname{argmin} d(m_j, C_k) \quad (5)$$

**Step 6:** For cluster center  $C_k$ , new center is computed for all intensity valas in equation (6), (7) and (8). The  $m_j$  in the equation represents the intensity pixel nearest to the cluster and  $N_j$  represents the number of pixels in an image having the intensity value  $m_j$ .

$$N_{ck} = \begin{cases} N_{ck} + (m_j * N_j) & \text{if } N\left(\frac{x}{m_j}\right) = C_k \\ N_{ck} & \text{otherwise} \end{cases} \quad (6)$$

$$d_{ck} = \begin{cases} d_{ck} + (N_j) & \text{if } N\left(\frac{x}{m_j}\right) = C_k \\ d_{ck} & \text{otherwise} \end{cases} \quad (7)$$

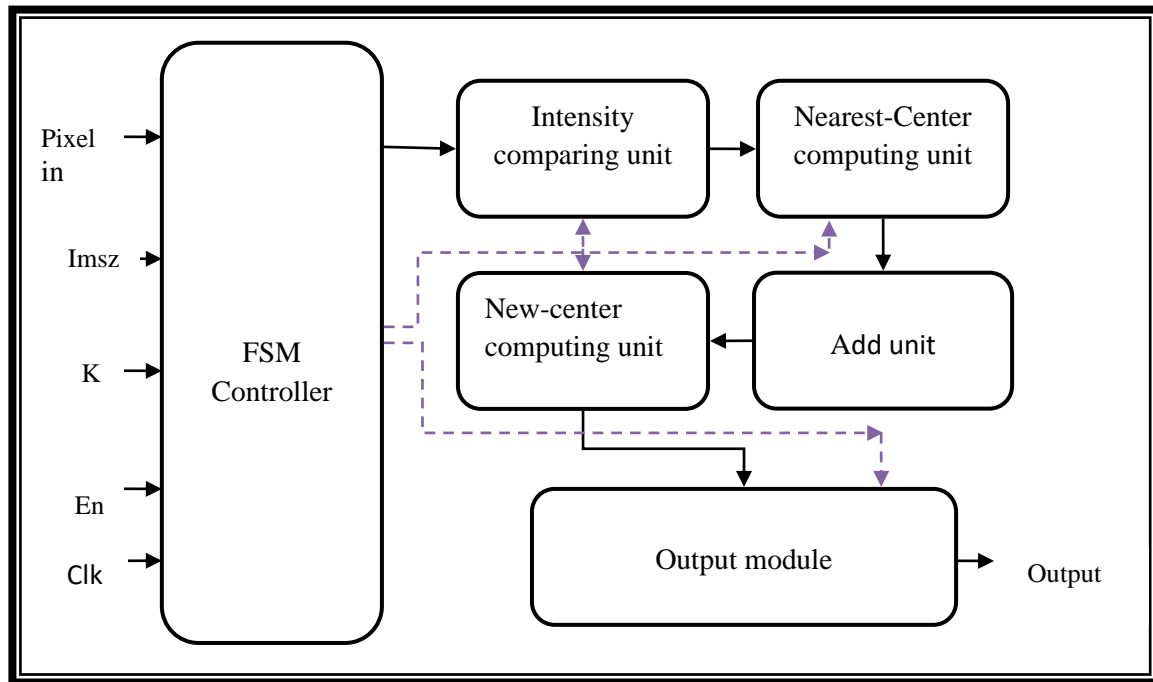
$$C_{k,new} = \frac{N_{ck}}{d_{ck}} \quad (8)$$

**Step 7:** Repeat steps 4 to 6 till the cluster center do not change its value.

The algorithm automatically stops if the cluster center does not change or if the maximum iteration has reached. Maximum of 10 iterations are fixed. After the algorithm converges, the image pixels are read again and the output for each image pixel read is the value of the nearest cluster center.

The architecture built using the above algorithmic steps to group the image pixels using K-Means algorithm is shown in Figure 1. The image given to the accelerator is pre-processed to remove any outliers as the K-means is sensitive to noise and would impact on the results of clustering. Hence an image free of noise or outliers is fed to the acclerator.

The input “Pixel-in” is an 8 bit image pixel to receive the input pixel serially in raster scan order. The size of a digital image is specified by a 24 bit “Imsz” input and hence the accelerator can support the comparison of the input pixel with the cluster center up to 16 mega pixel. The “K” input is a 5 bit input which specifies the number of groups and hence the image can be divided into maximum of 32 groups. Due to area constraints in the hardware, the accelerator can divide an image from 2 to 25 groups. The clock and enable signals used in the architecture are “En” and “Clk”. The dotted line indicates the control signal and solid line indicates the data signals. The “K” initial cluster centers are chosen randomly from an image. The subsequent sections explain the functionality and architecture formulated for each of the individual blocks used.



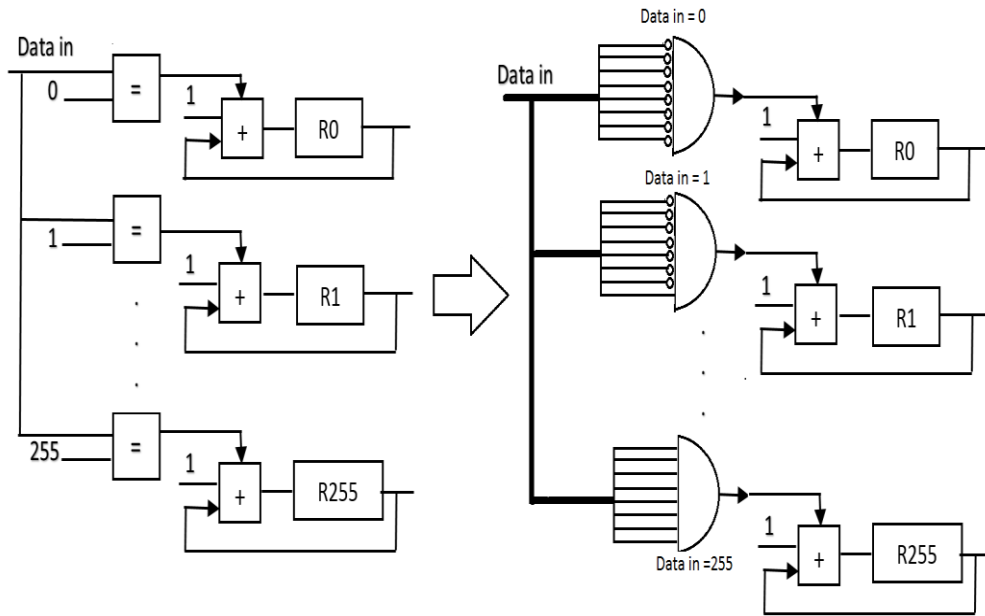
**Figure 1:** Architecture of the proposed algorithm

### 2.1. Intensity comparing unit:

The image pixel from “Pixel-in” is fed serially for each clock cycle and is compared against 256 possible intensity values, ranging from 0 to 255, corresponding to an 8-bit grayscale image. Registers R0 to R255 represent register which holds the frequency of these 256 intensity levels. The relevant register is incremented when the input pixel matches its corresponding intensity.

As observed, the comparator's output functions as an enable signal for the adder. When enabled, the adder increments the current value in the register by 1. The comparator compares the input with a specified value, reducing its complexity to an 8-bit AND gate, as illustrated in Figure 2. For instance, when comparing the input with the pixel value '0', the Data\_in input is inverted and sent to the 8-input AND gate. Conversely, when comparing with the pixel value '255', the Data\_in input is directly passed to the 8-input AND gate.

Initial cluster centers are selected randomly from the image. After reading the complete image, the count of each pixel intensities are available in the specified 256 registers. Hence, irrespective of the image size the architecture uses only 256 registers and 256 comparators reduced as AND gate.

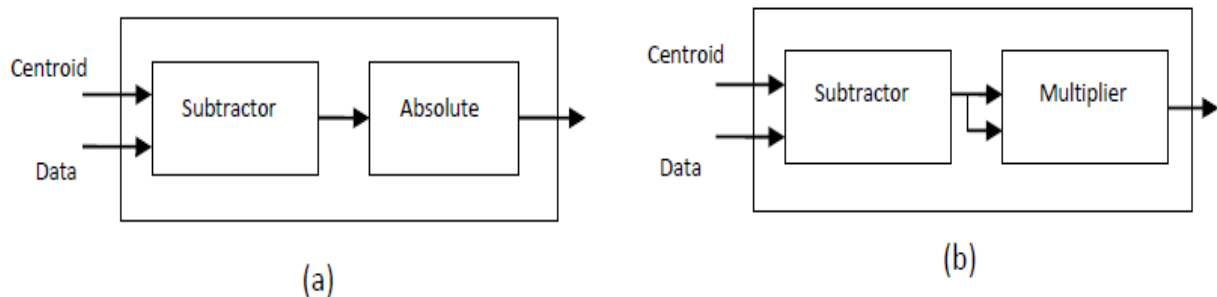


**Figure 2:** Intensity comparing unit

## 2.2. Nearest center computing unit:

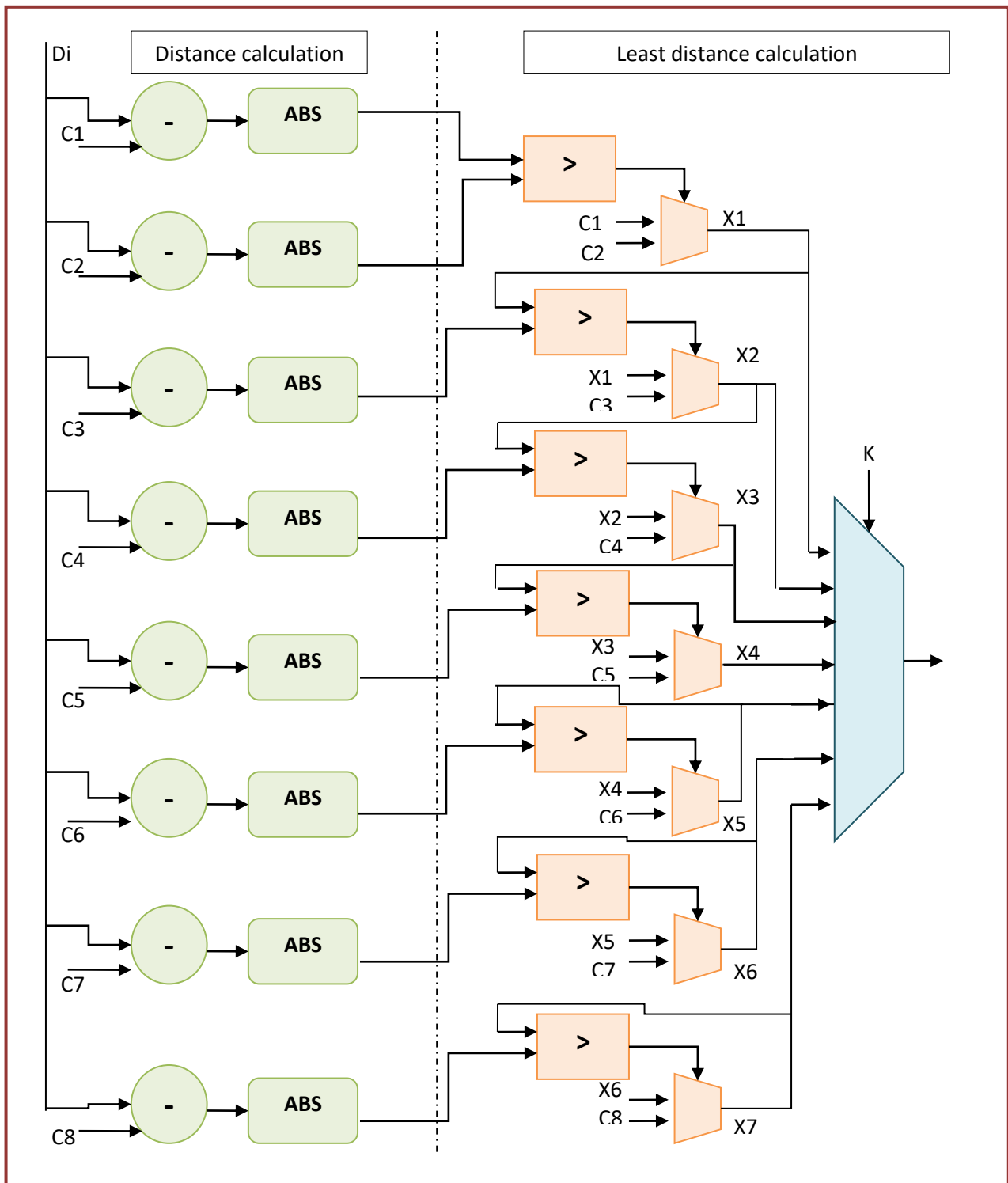
The intensity values of the pixel are compared with the cluster centers. The intensity value can span from 0 to 255. The accelerator supports maximum of 25 cluster centers that can be compared. Hence to compute the distance we require 6400 distance computing unit which calculates the distance in parallel between the intensity levels and the cluster center.

Two types of distance computing engines are examined in the architecture namely Absolute distance computer and squared distance computer as illustrated in Figure 3 represented by equation (3) and (4).



**Figure 3:** (a) Absolute distance computer (b) Squared distance computer

The nearest computing unit for computing the distance between an intensity pixel with 8 cluster centers is illustrated in Figure 4 as an example.



**Figure 4:** Nearest center computing unit shown with eight cluster centers

The input intensity pixel 'Di' is compared with all the eight cluster centers simultaneously to compute the distance. Later the computed distance is compared to select the least distance. Depending on the 'K' value, the nearest center is computed. The output of the nearest centre computing unit is the nearest cluster center value. For example, if 'K'=2, then the multiplexer selects 'X1'. The 'X1' contains the value of either C1 or C2. If distance between input intensity 'Di' is nearer to 'C1' then X1 contains 'C1' else it contains 'C2'. If 'K' is 3 then multiplexer selects the nearest cluster center as 'X2' which contains the nearest pixel center which may have the value 'C1', 'C2' or 'C3' depending on the distance between 'Di' and cluster center. The same concept is extended for comparing the input 'Di' with all

the eight clusters. Since, the 'K' input has 5 bits to support 25 groups, the 'Nearest center computing unit' has 24 comparators and 25 multiplexers to compare intensity with possible 25 cluster centers.

### 2.3. Add unit:

The output of the 'Nearest computing unit' is the corresponding nearest cluster center value for an intensity 'Di'. For each intensity value, the corresponding nearest center value is stored in the internal register. There are 256 registers each corresponding to an intensity level which hold the value of corresponding nearest cluster center. The 'Add unit' sums up all the pixel intensities which is nearer to the cluster center. Depending on the cluster centers, the pixel intensities nearer to all the cluster centers are summed up and stored in a register called Si. Also, the total number of pixels having same intensity ii is also stored in register 'Ci'.

The Pseudo code for the algorithm to compute addition is shown below

```

Algorithm 1:

Initialize s=0, c=0
for i<=0 to 255
{
  if(d(i)== center1)
  {
    s1= s1 + (i*reg(i))
    c1= c1 + reg(i)
  }
  elseif(d(i)== center2)
  {
    s2= s2 + (i*reg(i))
    c1= c2 + reg(i)
  }
  .....Continues for all the clusters
}
New cluster Center = s/c

```

The registers 'Si' corresponds to register having the sum of all intensities having same cluster center and 'Ci' correspond to total count of intensity value. For example, in a single cluster if there are ni pixels having intensity xi and mi pixels having intensity yi then the new cluster center is computed as shown in equation (3)

$$\text{New cluster center} = ((ni*xi) + (mi*yi)) / (ni + mi) \quad (9)$$

The division in equation (3) is computed using 24 bit divider whose Pseudo code is shown in Algorithm 2. Due to parallel hardware, the division operation is performed in single cycle.

```

Algorithm 2

Initialize temp_divisor = concatenation{1 bit of value, divisor, (n-1) bits of value 0}
temp_remainder = concatenation{(n-1) bits of value 0, dividend}
If (dividend=0)
{
  quotient= 0;
}
else
{
  For i <= 0 to (n-1)
  {
    temp_result = temp_remainder - temp_divisor;
    if(temp_result((2n-1) - i)=1)
      quotient(n-i) = 1'b0;
    else
    {
      Quotient (n-i) = 1'b1;
      temp_remainder = temp_result;
    }
    temp_divisor = temp_divisor >> 1;
  }
}

```



2.4. Output module:

The output module compares the previous cluster center with the present cluster center. If they are same, then the clustering operation stops else again the new cluster center is computed in the same procedure. The accelerator runs with maximum iteration of ‘10’. The entire operation is controlled by the ‘FSM controller’.

3. RESULTS AND DISCUSSIONS

The architecture for K-means was designed using Verilog HDL which is simulated in Xilinx I-Sim simulator and synthesized using Virtex 6 FPGA to examine the hardware characteristics in terms of area, timing and power. The cluster size was varied and the hardware utilization was examined.

The first step in the verification of the architecture is testing the clustering operation for various standard images. MATLAB tool was used to convert the images in jpeg format to binary format. These binary values of the image pixels were stored in an external memory and the image pixel was fed serially into the designed hardware accelerator. The clustering hardware accelerator designed clusters the image data into specified number of groups and for each of the pixel intensity it sends the nearest cluster center and stores it in an external memory for further analysis.

The clustered binary output data stored in binary format is again visualized using MATLAB for different values of K and the visual results obtained are depicted in Figure 5.

Original image				
Color clustering for K=15				
Color clustering for K=12				
Color clustering for K=10				






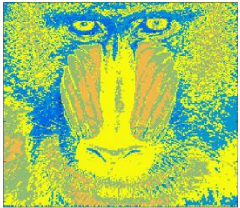
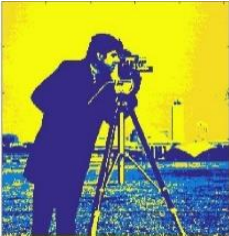

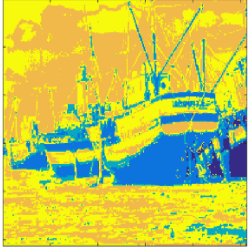


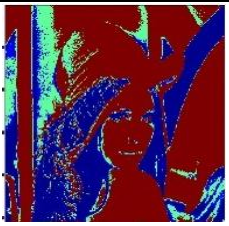





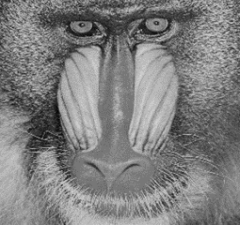






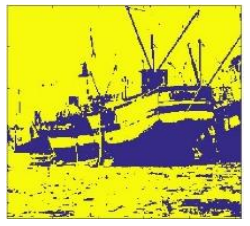

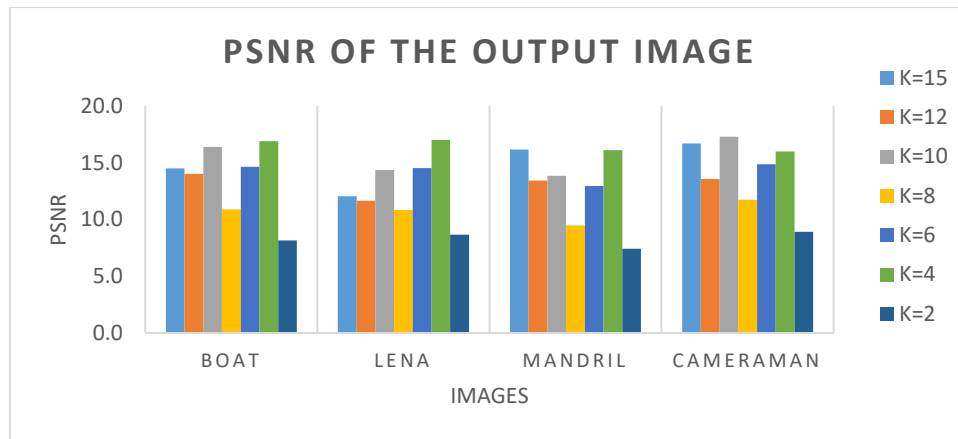
Color clustering for K=8				
Color clustering for K=6				
Color clustering for K=4				
Original Image				
Color clustering for K=3				
Color clustering for K=2				

Figure 5: Clustered output for varied value of K

The clustered output image visualized in MATLAB is analyzed for peak signal to noise ratio (PSNR) as illustrated Figure 6.



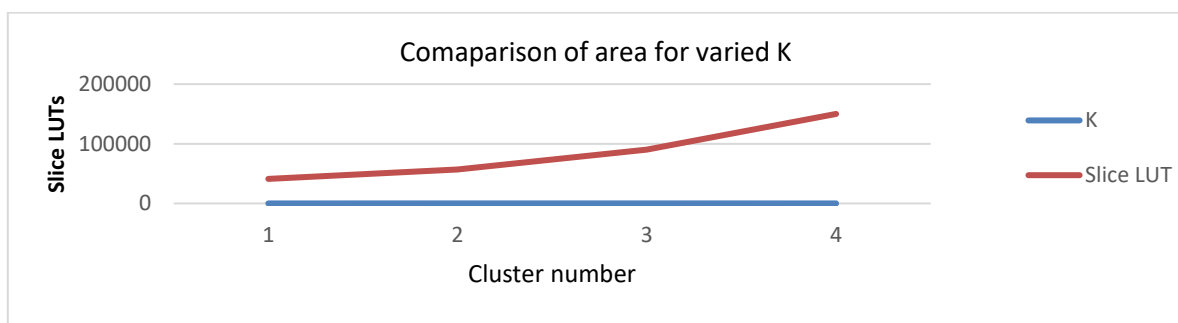
**Figure 6:** Obtained PSNR for various output images

The second part is to analyze the hardware characteristics of the proposed accelerator. For comparison, the original K-means algorithm that compares each of the pixel intensity with the cluster center was also implemented. Both the accelerators were synthesized using Virtex-6 FPGA and the area occupied is tabulated in Table 1

**Table 1:** Area comparison of original and proposed K-means accelerator

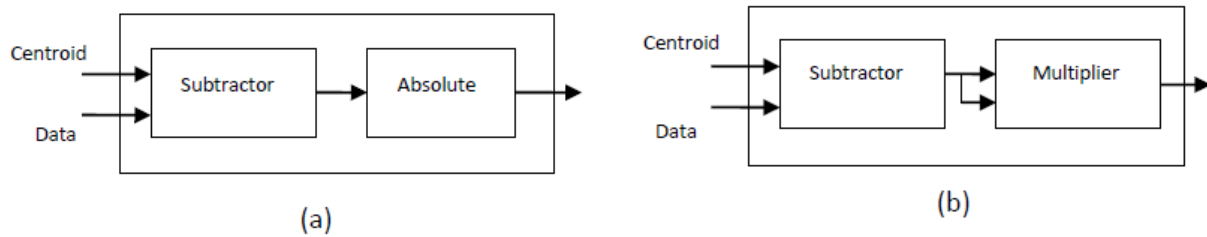
Logic utilization in Virtex 6 FPGA(XC6VLX75)	Original accelerator	Proposed Accelerator
Slice Registers	6767	6813
Slice LUTs	140098	41143
LUT-FF pairs	4615	2405
DSP block	236	2

As seen, the original accelerator occupies 3.4 times more slice LUTs compared to the proposed accelerator. Also to analyze the hardware cost, the number of grouping factor 'K' is varied and the plot is shown in Figure 7 where it is seen that the slice LUTs vary linearly with 'K'. This is because, 'Nearest center computing unit' requires more comparators and multiplexers as K is increased.



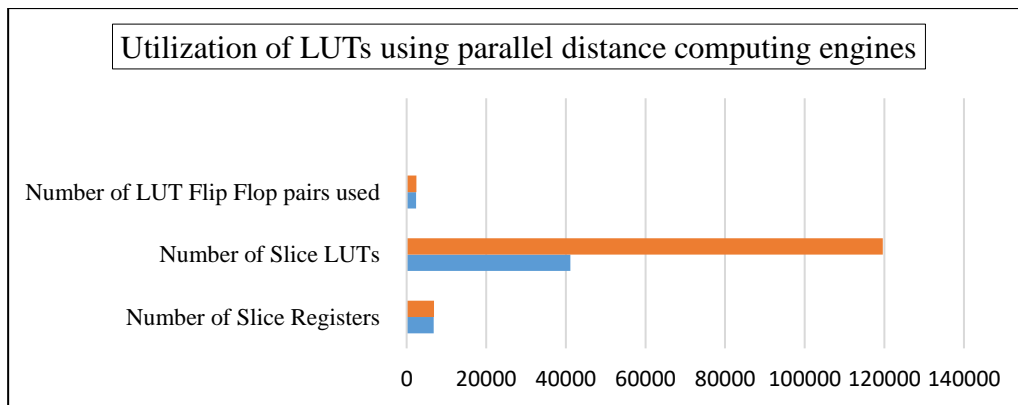
**Figure 7:** Variation of Slice LUTs with 'K'

The accelerator as mentioned earlier was tested for 2 types of distance computations. Parallel architectures using Absolute distance computer and squared distance computer was implemented. These distance computers compute the distance between the image pixel and the cluster centers in parallel using 256 computing engines where one of the computing engines is shown in Figure 8.



**Figure 8:** (a) Absolute distance computer (b) Squared distance computer

The hardware utilized by both the distance computing engines is depicted in Figure 9. The squared distance computer (Euclidean) as seen uses multiplier and hence occupies 3 times more area than absolute distance computer (Manhattan). Both the distance computer selects the nearest cluster center and hence the PSNR obtained for both the architecture remains same for the given image.



**Figure 9:** Hardware utilization of accelerators using two types of distance computers.

Next step is to analyze the number of clock cycles required to process the image of the given size. The accelerator proposed has five internal blocks namely 'Intensity computing unit', 'Nearest center computing unit', 'New center computing unit', 'add unit' and 'Output unit'. Except output unit, rest all of the blocks completes its operation in single clock cycle due to parallel hardware and hence 256 intensity levels of an image are processed simultaneously.

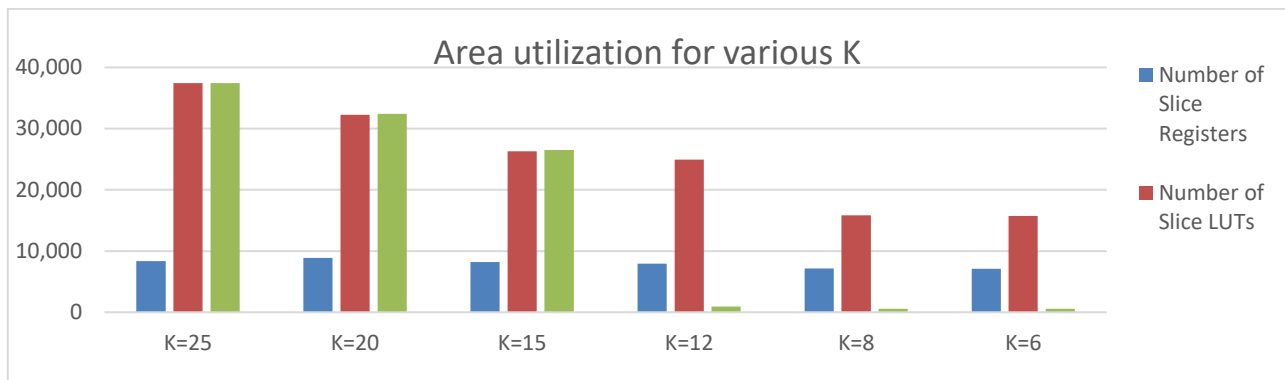
The intensity computing unit require the clock cycle depending on the image size. If the size of an image is 256 X 256, then it takes '65536' clock cycles to read all the pixel intensities of an image and compare it with the 256 intensity levels in parallel. After initial comparison, 'Nearest center computing unit', 'New center computing unit' and 'add unit' requires one clock cycle each for parallel computations. These units work iteratively till the cluster center is converged. The maximum iteration provided by the accelerator is 10 and hence maximum of 40 clock cycles are required if the clustered data is stored in internal memory for further processing. The accelerator operates at a clock frequency of 26.569MHz for all value of 'K' and hence the given 256 X256 image can be processed in 2.46 ms. Hence such 405 images can be processed in one second which clearly indicate that the proposed accelerator can be used in real time applications.

The proposed accelerator uses parallel architecture. Hence to reduce the area, the architecture was modified to accommodate the accelerator in low end FPGAs. Instead of using 256 parallel computations in 'Nearest center computing unit', if only one comparison is done with one intensity unit with all the cluster centers, then the unit requires 256 clock cycles to complete the operation. Now the total clock cycle required to complete the clustering operation is 68126 clock cycles which is divided as 65536 clock cycles for initial comparisons and totally 259 clock cycles for rest of the computations in other units. As the total maximum iteration is 10, it requires 2590 clock cycles to converge. The accelerator operates at a clock frequency of 26.569MHz for all value of 'K' and hence the given 256 X256 image can be processed in 2.56ms. Hence such 390 image frames can be processed in a second.

The area utilization using Virtex 6 FPGA with single and parallel hardware in 'Nearest center computing unit' is shown in Table 2 which depicts that the LUT utilization reduces from 88% to 31%. Also the area utilization for various 'K' using single computing unit is shown in Figure 10.

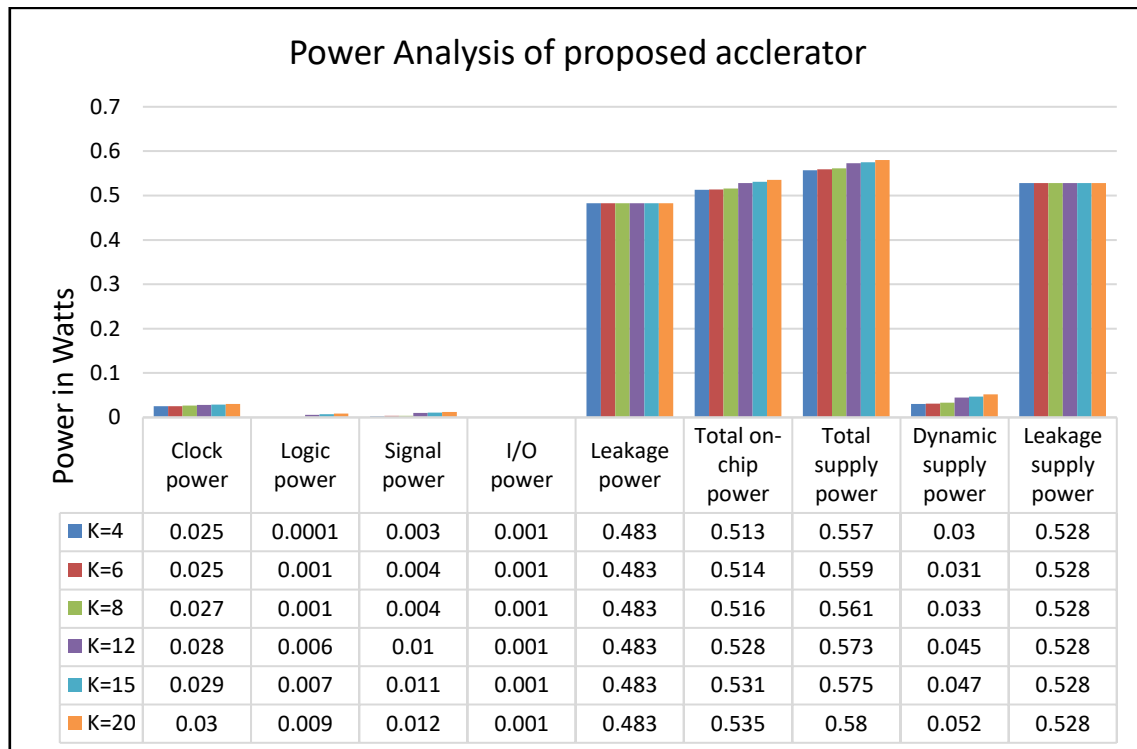
**Table 2:** Comparison of Logic utilization using Virtex 6 using single distance computation in nearest center computing unit'

Logic utilization in Virtex 6 FPGA(XC6VLX75)	With parallel architecture	Using single computing unit
Slice Registers	6813 (7% utilization)	6845 (7% utilization)
Slice LUTs	41143 (88% utilization)	14812 (31% utilization)
LUT-FF pairs	2405 (5% utilization)	2492 (5% utilization)
DSP block	2 (0.0001% utilization)	2 (0.0001% utilization)

**Figure 10:** Area utilization for various K using single distance computation in 'Nearest center computing unit'

As the utilization of area is less in Virtex-6, the accelerator was implemented in Spartan-3 FPGA. The accelerator when implemented on Spartan-3 FPGA operates at a clock frequency of 7.259MHz for all value of 'K' and hence the given 256 X256 image can be processed in 9.39ms. Hence such 106 images can be processed in a second.

The power dissipated by the accelerator is analyzed using Xilinx X-Power which is shown in Figure 11.

**Figure 11:** Power analysis of the designed accelerator for various K.

The comparison of the designed K-means hardware accelerator with the existing techniques is given in Table 3.

**Table 3:** Comparison of proposed accelerator with the literaturue

Implementation	Algorithm implemented	Fabric used for implementation	Operating frequency	Supported image size	No. of frames processed in seconds
Reference [3]	K-Means	Hardware software co design	40MHz	614x512	Not specified
Reference [6]	K-Means	FPGA- Virtex II	66 MHz	Upto 768x512	20-30 fps
Reference [10]	K-Means with automatic K selection	ASIC-90nm	233 MHz	320X240	1 frame takes less than 2 sec
Reference [11]	Hierarchical K-Means	ASIC-90nm	333 MHz	320X240	18fps when K=16
Reference [12]	Online clustering	ASIC-90nm	400 MHz	256X256	8 fps
Reference [16]	Mean shift	FPGA- Stratix III	Not specified	Upto 300K	Not specified
Reference [17]	Mean shift	FPGA	125MHz	128 samples	Not specified
Implementation	Algorithm implemented	Fabric used for implementation	Operating frequency	Supported image size	No. of frames
Reference [18]	Fuzzy C-Means	Simulation	Not specified	185X185	<1
Reference [19]	Mean shift	FPGA- Spartan 6	122MHz	300X420	351
Reference [20]	K-Means	Xilinx Zedboard	150MHz	640X480	82
Proposed Accelerator	K-Means	FPGA Virtex 6	26.6MHz	Upto 16Mp	405 fps for an image size of 256X256

The proposed hardware accelerator can process more images compared to the existing literature even though the operating frequency is less.

#### 4. CONCLUSION:

The proposed hardware accelerator for clustering and image using K-Means clustering can process 256 X2 56 images at 405 frames per second, outperforming other accelerators while requiring less area. The newly proposed method can handle images up to 16MP using less area and hence can be implemented using low end FPGA significantly reducing the cost of embedded systems used for image analysis. However more robust accelerator can be built which would automatically cluster itself into the optimum groups.



## REFERENCE:

- [1] J. MacQueen, "Some methods for classification and analysis of multivariate bservations," in Proc. 5th Berkeley Symp. Math. Stat. Probab., 1967, pp. 281–297.
- [2] Kanungo, Tapas, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. "An efficient k-means clustering algorithm: Analysis and implementation." *IEEE Transactions on Pattern Analysis & Machine Intelligence* 7 (2002): 881-892.
- [3] A. G. d. S. Filho, A. C. Frery, C. C. de Araújo, H. Alice, J. Cerqueira, J. A. Loureiro, M. E. de Lima, M. d. G. S. Oliveira, and M. M. Horta, "Hyperspectral images clustering on reconfigurable hardware using the K-means algorithm," in Proc. Symp. Integr. Circuits Syst. Des., Sep.2003, pp. 99–104.
- [4] Estlick, Mike, Miriam Leaser, James Theiler, and John J. Szymanski. "Algorithmic transformations in the implementation of K-means clustering on reconfigurable hardware." In *Proceedings of the 2001 ACM/SIGDA ninth international symposium on Field programmable gate arrays*, pp. 103-110. ACM, 2001
- [5] V. Bhaskaran, "Parametrized Implementation of K-means Clustering on Reconfigurable Systems", M.S. thesis, Dept. Elect. Eng., Univ. of Tennessee, Knoxville, TN, 2003.
- [6] T. Maruyama, "Real-time K-Means clustering for color images on reconfigurable hardware," in Proc. Int. Conf. Pattern Recog., 2006, pp. 816–819.
- [7] Boris Maliatski and Orly Yadid-Pecht. " Hardware-Driven Adaptive k-Means Clustering for Real-Time Video Imaging", *IEEE transactions on circuits and systems for video technology*, volume 15, no 1(2005)
- [8] T.-W. Chen, C.-H. Sun, J.-Y. Bai, H.-R. Chen, and S.-Y. Chien, "Architectural analyses of K-Means silicon intellectual property for image segmentation," in Proc. IEEE Int. Symp. Circuits Syst., May 2008, pp. 2578–2581.
- [9] T.-W. Chen and S.-Y. Chien, "Bandwidth adaptive hardware architecture of K-means clustering for video analysis," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 6, pp. 957–966, Jun. 2010.
- [10] Tse-Wei Chen, Chih-Hao Sun, Hsiao-Hang Su, Shao-Yi Chien, Daisuke Deguchi, Ichiro Ide, Hiroshi Murase" Power-efficient hardware architecture of K-means clustering with Bayesian-information-criterion processor for multimedia processing applications" , *IEEE Journal. Circuits and system.* vol. 1, no. 3, pp. 357-368, Sep. 2011.
- [11] Tse-Wei Chen and Shao-Yi Chien, Member, IEEE" Flexible Hardware Architecture of Hierarchical K-Means Clustering for Large Cluster Number". *IEEE transactions on very large scale integration (VLSI) systems*, vol. 19, no. 8, august 2011.
- [12] Tse-Wei Chen and Makoto Ikeda" Design and Implementation of Low-Power Hardware Architecture With Single-Cycle Divider for On-Line Clustering Algorithm" *IEEE Transctions on Circuits and system.*, vol. 60, no. 8, pp. 2168-2175, Aug. 2013
- [13] Hussain, Hanaa M., Khaled Benkrid, Huseyin Seker, and Ahmet T. Erdogan. "Fpga implementation of k-means algorithm for bioinformatics application: An accelerated approach to clustering microarray data." In *2011 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, pp. 248-255. IEEE, 2011.
- [14] Deng, Tiantai, Danny Crookes, Fahad Siddiqui, and Roger Woods. "A New Real-Time FPGA-Based Implementation of K-Means Clustering for Images." In *Intelligent Computing and Internet of Things*, pp. 468-477. Springer, Singapore, 2018.
- [15] Farivar, Reza, Daniel Rebolledo, Ellick Chan, and Roy H. Campbell. "A Parallel Implementation of K-Means Clustering on GPUs." In *Pdpta*, vol. 13, no. 2, pp. 212-312. 2008.
- [16] Craciun S, Kirchgessner R, George AD, Lam H, Principe JC. A real-time, power-efficient architecture for mean-shift image segmentation. *J Real-Time Image*, Springer, Proc. 2014
- [17] CH Tsai, HH Lee, WJ Yu, CY Lee "A 2 GOPS quad-mean shift processor with early termination for machine learning applications" *IEEE Transctions on Circuits and system*, vol. 60, no. 8, pp. 2168-2175, Aug. 2014
- [18] Raval, Khushbu, Ravi Shukla, and Ankit K. Shah. "Color Image Segmentation using FCM Clustering Technique in RGB, L\* a\* b, HSV, YIQ Color spaces." *European Journal of Advances in Engineering and Technology* 4, no. 3 (2017): 194-200
- [19] Tehreem, Amna, Sajid Gul Khawaja, Asad Mansoor Khan, Muhammad Usman Akram, and Shoab A. Khan. "Multiprocessor architecture for real-time applications using mean shift clustering." *Journal of Real-Time Image Processing* (2017): 1-14.



- 
- [20] Deng, Tiantai, Danny Crookes, Fahad Siddiqui, and Roger Woods. "A New Real-Time FPGA-Based Implementation of K-Means Clustering for Images." In *Intelligent Computing and Internet of Things*, pp. 468-477. Springer, Singapore, 2018.
  - [21] K. Makarychev, A. Reddy, L. Shan, Improved guarantees for K-means++ and K-means++ Parallel, *Adv. Neural Inf. Proces. Syst.* 33 (2020) 16142–16152.
  - [22] Y. Mao, D. Gan, D.S. Mwakapesa, Y.A. Nanekaran, T. Tao, X. Huang, A MapReduce-based K-means clustering algorithm, *J. Supercomput.* 78 (4) (2022) 5181–5202.
  - [23] P. Olukanmi, F. Nelwamondo, T. Marwala, B. Twala, Automatic detection of outliers and the number of clusters in k-means clustering via Chebyshevtype inequalities, *Neural Comput. & Applic.* 34 (8) (2022) 5939–5958.
  - [24] Zhu, Z., & Liu, N. (2021). Early warning of financial risk based on K-means clustering algorithm. *Complexity*, 2021.
  - [25] Zhuang, Y., Mao, Y., & Chen, X. (2016). A limited-iteration bisecting K-means for fast clustering large datasets. In *2016 IEEE Trustcom /BigDataSE /ISPA*, 2257-2262.
  - [26] M. Zubair, M.D. Iqbal, A. Shil, M.J.M. Chowdhury, M.A. Moni, I.H. Sarker, An improved K-means clustering algorithm towards an efficient datadrivenmModeling,, *Annals of Data Science 2022* (2022), <https://doi.org/10.1007/s40745-022-00428-2>.
  - [27] A. Abernathy, M.E. Celebi, The incremental online k-means clustering algorithm and its application to color quantization, *Expert Syst. Appl.* 207 (2022) 117927.
  - [28] K. Abhishekkumar, C. Sadhana, Survey report on K-means clustering algorithm, *Int. J. Mod. Trends Eng. Res* 4 (2017) 218–221.
  - [29] L. Abualigah, A. Diabat, Z.W. Geem, A comprehensive survey of the harmony search algorithm in clustering applications, *Appl. Sci.* 10 (11) (2020) 3827.
  - [30] Ikotun, Abiodun M., Absalom E. Ezugwu, Laith Abualigah, Belal Abuhaija, and Jia Heming. "K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data." *Information Sciences* 622 (2023): 178-210.