Cumulative Mean Intensity Differential Transition Algorithm for Edge Detection

Ganesh Pai, M Sharmila Kumari

Abstract— ***Edge Detection plays a vital role in machine vision applications and thereby variety of edge detection algorithms being developed over time for both grey scale and colour images. In this paper, a new technique for edge detection called cumulative mean intensity differential transition algorithm (CuMIDT Algorithm) is proposed. This approach focuses on learning variations in the local pixel intensities and predicting the possible edge when the intensity deviation goes out of the stipulated window area. Ramps at the edge boundaries and zero crossing are addressed using differential transition model. Experimentation are done on standard FDDB dataset and real dataset. It is observed that the proposed approach gives better results when compared to the recently proposed novel edge detection algorithms.***

Keywords: *Edge Detection, CuMIDT, Differential transition model*.

I. INTRODUCTION

Core of any machine vision algorithm is object detection in image processing which ultimately hails towards edge detection. Precisely detected edges plays a key role in the accurate detection and recognition of objects. Achieving precise edge detection for a complex structures and patterns with poor illumination, blur images is a challenging task.

Earlier works on edge detection dates back from 1965 through computation of the first order derivative of the input image using Robert's, Prewitt's and Sobel's operators [1]. Rotation invariant Laplacian isotropic filter using second order derivatives was further refined by Marr and Hildereth as LoG function [1] and further improved by Canny [2]. Subpixel based edge detection method are discussed in [3][6][7]. Fast Image Edge Detection based on Faber Schauder Wavelet and Otsu threshold [4] and Neutrosophic Set structure using maximum norm entropy [5] are found to be the recent developments. Neural network based models for edge detection were used in [8][9][11]. A non-iterative approach that identifies the local maxima of the normalized absolute values of the RBF interpolant coefficients for detection is used in [10].

II. PROPOSED METHODOLOGY

In this section, our proposed methodology for edge detection called cumulative mean intensity differential transition algorithm (CuMIDT Algorithm) is presented. In this approach, the intensity variation within a region is captured and the regional average intensity is computed. The borders of these aggregated regions forms the factor for detection of a possible edge. The entire methodology is developed as a five step process containing pre-processing, computation of cumulative mean, differential transition model, binarization of transition points, local pattern correction and noise elimination.

Revised Manuscript Received on January 31, 2020.
***** Correspondence Author

Ganesh Pai*, Department of Computer Science & Engineering,P. A. College of Engineering, Mangalore, India.Email: ganeshpai24@gmail.com

M SharmilaKumari, Department of Computer Science & Engineering, P. A. College of Engineering, Mangalore, India. Email: sharmilabp@gmail.com

A. Pre-processing

Given a grayscale image, it is subjected to three step process. Smoothing, sharpening (optional step), and normalization. The image is first smoothened using 2D Gaussian function. If x and y are the coordinates of the pixel in an image and σ is the standard deviation of the Gaussian function, then the 2D Gaussian function G(x,y) is given by

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad (1)$$

With *I* as the input image and G as the Gaussian function, Gaussian smoothened image I_g is given by

$$I_g = I * G \qquad (2)$$

The smoothened image I_g is then optionally sharpened. While smoothing, certain edge areas with enough intensity to distinguish as an edge gets blurred outcausing performance degradation. To overcome this, the image is optionally sharpened using a 3x3 sharpening filter. This will improve the pixel intensities at the edges and thus improving the detection rate. Equation (3) represents the sharpening filter kernel *F*that can be used. Sharpened imageI_s (4)is obtained by convolving image$I_g$with the kernel *F*.

$$F = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix} \qquad (3)$$

$$I_s = I_g * F(4)$$

Filtered image is then normalized and represented as I_n

$$I_n = \begin{cases} \frac{I_s}{255}, \ when\ sharpening\ is\ applied \\ \frac{I_g}{255}, \ when\ sharpening\ is\ not\ applied \end{cases} \qquad (5)$$

B. Computation of cumulative mean

Initially, a threshold value of *τ*(window area *W = 2τ*) is defined within which the differential value should range. Starting from the first pixel of the first row, assuming as i^th pixel, a cumulative mean m_c of the pixel is computed which is initially the pixel value.

$$m_c = \frac{1}{i-y_s} \sum_{y=y_s}^{i-1} I(x,y)(6)$$

wherei>0 and y_s=0 initially for horizontal computation

Next, the difference between the cumulative mean m_c of i^th pixel and the (i+1)^th pixel is computed.

Ifthe modular difference is less than or equal to threshold τ, $(i+1)^{th}$ pixel is considered for mean computation and progressed to the next pixel. If the difference is greater than the threshold value, all the pixel values considered under current mean computation up to *i*is set to the value of m_c and repeat the process with the $(i+1)^{th}$ pixel value. For an m×n image, this technique is applied for each m rows. The algorithm is as follows.

```
Algorithm CumulativeMean($I_n$):     [Normalized input image]
Step 1: Set $m_c \leftarrow I_n(0,0)$
Set $\tau \leftarrow W/2$                    [W is the window size]
Step 2: for each row x:
$y_s \leftarrow y_c \leftarrow 0$                  [$y_c$ is count of pixels]
for each column y:
if $|m_c - I_n(x, y)| \geq \tau$:
$XI_n(x, y_s to y) \leftarrow m_c$
$y_s \leftarrow y_c \leftarrow m_c \leftarrow 0$
$m_c \leftarrow sum(I_n(x, y_s to y)) / y_c$
$y_c \leftarrow y_c + 1$
```

This will capture the pixel intensities that range within a stipulated window area as one intensity since all those pixels are representations of the same region or image object at an abstract level. The selection of cumulative mean value as a representation of the intensity group is trivial as it can be replaced by other parameter such as median or mode. These varying intensity groups creates a block representation of the image. The intensity change that appears between the blocks is the feature identified as a possible existence of an edge.
This approach computes pixel intensity variations in horizontal direction only and hence is able to capture the vertical and diagonal edges but not horizontal edges. To capture horizontal edges, the algorithm is reapplied in the vertical direction by changing horizontal variations to vertical variations. Computation of cumulative mean for vertical direction is done through eq. (7):

$$m_c = \frac{1}{i-x_s}\sum_{x=x_s}^{i-1} I(x,y) \quad (7)$$

wherei>0 and x_s=0 initially for vertical computation

Let XI_n and YI_n be the cumulative intensities computed in the horizontal and vertical directions respectively.

C. Differential transition model (DTM)

In the computation of the cumulative mean, the pixel intensities close to each other within the window will share a common value. As we approach near an edge, there will be a shift in the intensity level.Thus subsequent pixel intensities may not be falling within the window. This will create an intensity ramp in these regions. The DTM is basically intended to identify such intensity ramps and converts them into an intensity step. The regions with constant intensities are left untouched. In the region of intensity ramp, possible existence of certain constant intensities in between or at the corners are too addressed here.

The DTM is developed using a state transition diagram. The diagram is incorporated with seven states with two final states in two distinct situations. Fig. 1 and Table 1 shows the state transition diagram and its table respectively. Difference *d*is computed between the *i*th and *(i-1)*th pixel i.e. $d = XI_n(x, y_i) - XI_n(x, y_{i+1})$. The outcome of the difference are zero, positive or negative. The difference will be zero for constant intensities, positive for transition from lower intensity to a higher intensity, and negative for transitions from higher intensity to lower intensity. These values are the input to our state machine. State model also addressed situations with sudden transition from positive to negative i.e. zero crossings.
As there are two cumulative mean images XI_n and YI_n received as input to this stage, the DTM is applied to both the images separately. The model is applied to each row of XI_n and each column of YI_n. The outcome is an image with all ramp regions converted to step and represented as transformed images XI_t and YI_t respectively.

The differential state transition diagram of Fig. 1 presents various transitions that occurs while traversing over image pixels. Each transition is labeled with two information. The first entry is the input the state receives and the second is the entry labeled **A** which represents the core action performed when the corresponding input is received. The state model uses two variables namely *cnt*, representing each of the transition that occur, 0 to +ve, 0 to -ve, +ve to +ve , –ve to –ve, +ve to –ve and –ve to +ve and the second variable keeps track of count of pixels with constant intensities along the ramp. This is represented by *ztc* called zero transition count. This value is monitored by a constant value named *mztc* called maximum zero transition count. If the *ztc* exceeds *mztc*, it is considered as a stepping state and necessary intensity adjustment is done. State 2 and state 4 are basically representing these stepping states or accepting state of the state diagram. In the stepping state, the intensity ramp can be converted to a step, using different methods. The method chosen in this work is the average of the pixels intensities at both ends. All the pixels less than or equal to the average value are set with a pixel value at left most end of the intensity ramp and all the pixels greater than the average value are set with a pixel value at right most end of the intensity ramp. Alternately, median or mean of the pixel intensities can also be considered. The effect of each varies based on the local intensity value in the image. This is followed by resetting all the counters. When a +ve input is received in state 1 and 5, it makes a transition to state 2, converts the ramp to step for all pixels encountered till that point, resets the counters and switches to state 3. This is represented by dashed transition line in Fig. 1. Similar action is taken when a +ve input is received at states 3 and 6. States 2 and 4 are not going to receive any inputs, rather they are intended to do a stipulated task and make a blank transitions accordingly based on the previous transitions. Table 1 shows this transition as no input transition.

TABLE1. State Transition Table

State\ Input	0	+ve	-ve	No input	State\ Input	0	+ve	-ve	No input
0	0	3	1	-	4	-	-	-	0/1
1	5	2	1	-	5	5	2	1	-
2	-	-	-	0/3	6	6	3	4	-
3	6	3	4	-					

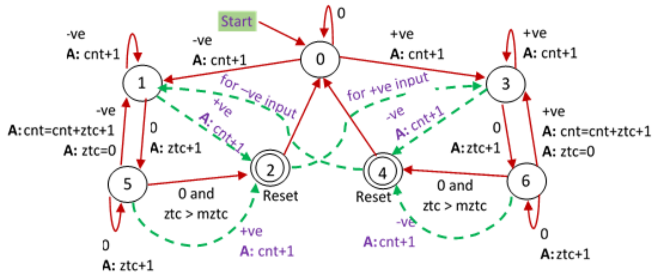

Fig. 1 Differential state transition diagram

D. Binarizing Transition Points

The transformed image $XI_t$and $YI_t$are now ready to be unified and binarized. In this step, transition points in $XI_t$and $YI_t$are searched and represent as a binary image with values 0 or 255. The pixel positions where there are no transitions found are represented as 0 and pixel positions where we find a step are represented by 255. The outcome here is a binary image I_b with transitions from both $XI_t$and YI_t. If β is the function that takes $XI_t$and $YI_t$and returnsa black and white image I_b, then,

$$I_b = \beta(XI_t, YI_t) \quad (8)$$

I_b now contains information about horizontal, vertical and diagonal edge pixels.

E. Local pattern correction and noise elimination

In an ideal case, I_b will contain all the pixels exactly representing all the edges of the original image. But in real, we may not get an exact edge due to the local intensity variations and poor illuminations of the image. Certain edge points may not be captured that are visible at an abstract level but loose the importance at the pixel level due to its surrounding pixel intensities.

As binarized image I_b contains information from two distinctly computed pixel transitions, while integrating, there may be certain edge points detected at nearby distinct positions given by $XI_t$and YI_t rather than at same position, due to the local intensities calculated in different directions, thereby edges getting misaligned. On the other hand, there are situations at some edge locations, the pixel transition may not be detected, again due to local intensities values. Such edge pixel gets shifted to a distant location away from the edge. These are considered as noisy points that need to be eliminated.

Local pattern correction is basically realigning the misaligned edge points. The correction is done using a 3x3 pattern matrix. The main task done here is the prediction of the possible missing transition pixel and adding such pixel and prediction of possible unimportant/less important edge pixel that may not contribute much to the edge formation and thereby suppressing such pixels. It may not be exactly possible to do the corrections to all the misaligned pixels but a best effort is done to possibly correct with a 3x3 pattern.

To implement Local pattern correction, 106 pattern of size 3x3 each represented as a vector are developed which are the representation of the possible patterns that needs correction and a corresponding correction vector is developed to each of these. These patterns capture corrections to be done in all 8 directions (0 - 360^0 in steps of 45^0). Ten such correction patterns are as shown in Table 2. The matrix representation of the pattern vector is matched with each 3x3 matrix of the image to find a match, and applied with the correction matrix derived from the corresponding correction vector for each match found. All 106 patterns are checked for its existence and necessary corrections are applied. The 3x3 image matrix is read in the row order form from the image and represented as a patterns vector and mapped to its corresponding correction vector. The pixel value 255 is represented as 1 in the patterns. If ψ is the local pattern correction function then

$$I_{pc} = \psi(I_b) \quad (9)$$

Applying the local pattern correction will correct the edges up to some extent. Apart from the corrections, there is also a need to eliminate the noisy pixels emerged in the image. These noisy pixel are eliminated using a mask of varying size. The intension here is to identify and eliminate isolated pixels that are not part of the edges. There may be certain closely located noisy pixels. These are eliminated using larger mask size. The mask ensures that the noisy pixels are not connected to the main edge in any directions. M is one such mask of size 5x5. B is a XOR of the 5x5 pixel values of the image and the mask M.

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} B = I_{pc}|_{5x5} \oplus M \quad (10)$$

If any value of B is true, then there is a connection from any pixel of the inner 3x3 matrix with the outer pixel which may be a possible edge. In such case, move forward to the next pixel. If none of the values of B is true then, reset the inner 3x3 matrix of the image I_{pc} to 0. This can be done for varying size mask. The result of this is the isolated noisy pixels gets eliminated up to some extent, if not all. Some noisy pixel group will be of varying size and pattern. Different approach need to be developed to complete elimination of such noisy pixel groups. The resultant noise eliminated image, I_N, formed after applying the mask M to I_{pc} is the final edge image that can be projected. If ϕ is the noise elimination function, then

$$I_N = \phi(I_{pc}) \quad (11)$$

III. IMPLEMENTATION DETAILS

The CuMIDT algorithm is applied on a FDDB and real dataset of varying size grayscale images and implemented using Python and OpenCV library.

In the pre-processing stage, smoothing is done using Gaussian blur function with kernel of size 3x3 and standard deviation 0. This is followed by a normalization process. Cumulative mean is computed from the first pixel of the image with τ = 0.1, as it is observed as a favorable value for the chosen dataset. For poor illuminated images, τ need to be reduced and can be increased for images with good illumination. By setting τ = 0.1, window w = 0.2 thereby providing a span of 10% on either side of the zero mean. Here pixel intensity is allowed to vary by 25 levels on either side or an overall of 51 levels. Pixels varying by 51 levels are aggregated by its mean value. This process is applied in both horizontal and vertical directions to get two distinct cumulative images XI_n and YI_n.

DTM is applied on XI_n and YI_n separately to eliminate ramps in the image by converting them to step. The maximum zero transition count, *mztc*, is chosen to be 3 in our experimentation. *mztc* can be varied based on size of the image. In case of image with large dimension, 3 to 5 is an acceptable range.

With *mztc* set to 0 causes zero transition count not to be considered which is favorable in lower resolution images but will produce poor results in higher resolution images. Hence *mztc* need to be adjusted accordingly. This is followed by generating a binary the image at the transition points. Locations where block transitions occur are marked as white pixel. This is followed by local pattern correction and noise elimination. Here 106 patterns are used for applying the corrections. Noise eliminations can also make use of varying sized masks. 5x5 size masks is used as shown in previous section. The results are as shown in the next section.

IV. EXPERIMENTAL RESULTS AND COMPARITIVE ANALYSIS

With the grayscale input image in Fig. 2, Fig. 3 to Fig. 10 shows output of various stages. Fig. 3 shows the result of pre-processing without sharpening the image. In Fig. 4 and 5, one can observe the blocks of grey colours appearing in both horizontal and vertical directions respectively, due to the application of the cumulative average to all pixels falling in the window area. Pixels intensities within the window of 51 pixels (τ=0.1, w=0.2) are set with a representative value based of the actual local intensities. Fig. 6 and 7 are after applying DTM and converting all ramp regions to step. At a high level, we may not be able to clearly view this effect. Hence this effect is shown at pixel level in Fig. 11 to 14.

In Fig. 11 and 12, it is observed, the changes that occur at the edge transition points. If this is not done, then multiple edges detected at the edge areas, which results in false detection. Hence this step overcomes these false detections and considers them as one edge. Fig. 13 and 14 shows the effect of major role played by *ztc* by shrinking thick edge areas to thin edges in XI_n and XI_t respectively. In the bottom region of Fig. 13, we can find multiple pixels with same intensity within the edge ramp. These pixels with constant intensities along the ramp are considered as zero transition count (*ztc*) and are eliminated that are within maximum *ztc (mztc)*. Fig. 14 shows all such occurrences are eliminated.

Fig. 8 shows the result of unification of XI_t and YI_tastransition points and Fig. 9 after pattern correction and noise elimination. It can observed that complete noise elimination could not be done as mentioned earlier. Alternate noise elimination techniques can be further applied to fine tune noise elimination. Fig. 10 finally is the detected edges superimposed over the original image. Observe that almost all of the edges are detected. In addition, certain lines can be seen in places where in fact no sharp edge exists. This is due to the deviation of the pixel intensities from the cumulative mean at the local region. This will result in development of transition points that are later interpreted as edges. The possible solutions to overcome such situation is to carefully calibrate the τ value so that true edges does-not go undetected and false edges are not detected. Alternately, the work can be further extended by developing weighted transition points and points with more weights can be retained at the end as sharp edges.

Fig. 15 to 23 shows results of three more sample images with Fig. 15, 18 and 21 being the original images, Fig. 16, 19 and 22, their edge detected binary image and Fig. 17, 20 and 23 being the superimposed edges respectively. All the results here are achieved with threshold τ = 0.10 and no sharpening done while pre-processing. By slightly adjusting the value of τ, we can have certain level of fine tuning done. But increasing it or decreasing threshold value in large scale results in weakly detected edges.

Performance of the algorithm is proportional to the dimension of the image. As the implementation done here are in stages, there is a considerable delay seen due to multiple iterations of the image. Computation time will be substantially reduced with pipelined and parallel implementations. Conversely, implementing using C or C++, can gain a considerable performance boost. Table 3 shows the computation time for the various images at each stage when executed on Intel Core i3-3220 3.30GHz processor, 4GB RAM and 64-bit Linux OS.

The time shown in the Table 3 are average time of multiple iterations. Column 4 shows the time taken for computing the cumulative mean, column 5 for transforming the edge ramps to steps, column 6 for binarizing the transition points, column 7 for local pattern correction, column 8 for removal of noise and column 9, the total time consumed. As image resolutions varies, time consumed too varies. To incorporate relative time consumed, column 10 common metric where all timings are normalized to time per lakh pixels. A better approximation of the time taken for each image is observed. Average time taken per lakh pixels is 1.032s with a standard deviation of 0.204.

Another major observation is that noise elimination is the one which consumes the maximum time among all. Performance boost can be expected if we somehow reduce noise elimination time.

By excluding the noise elimination time, we get average time taken as 0.697s and standard deviation of 0.097 which is a considerable improvement over the previous results.

Fig. 26 to 32 shows various effects of sharpening the image and calibration of τ to control the detected edges. Fig. 25 shows the preprocessed image of the original image in Fig. 24 with sharpening applied. In Fig. 25, intensities of certain pixels at the edge areas has increased due to sharpening and thereby increasing the width of the edge. This can be seen in Fig. 33 before sharpening and Fig. 34 after sharpening. In Fig. 34, observe formation of thick black edge. Due to the formation of these thick lines in multiple regions in the image, we get a scattered edges or multiple edges being detected along the edge areas, as shown in Fig. 26 when τ = 0.10. A moderate correction to this can be applied by calibrating the τ value. Fig. 27, 28 and 29 shows results with τ = 0.20, τ = 0.25 and τ = 0.30 respectively and Fig. 30, 31 and 32 shows the result when τ = 0.20, τ = 0.25 and τ = 0.30 respectively. Certain edges may get missed as observed in Fig. 29 when τ = 0.30.The upper edge of the cap is barely detected but clearly detected when τ = 0.25 in Fig. 28. In case of his ears, excess lines are seen when τ = 0.10 but that is rectified at τ = 0.25 and starts diminishing as τ moves above 0.25 and some lines of the ears are disappeared at τ = 0.30.

The proposed algorithm is compared with sobel's operator with kernel size of 3x3, scale of 1 and delta value equal to 0 and canny edge detection algorithm with sigma value 50, kernel size 3x3, weak pixel value 1, strong pixel value 255, lower threshold of 0.05 and higher threshold of 0.15.The edges detected using these techniques and proposed technique are shown in Fig. 36 to 38 for the original image in Fig. 35 and Fig. 40 to 42 for the original image in Fig. 39 respectively.

The proposed CuMIDT algorithm is applied with threshold value, τ of 0.20 with sharpening Fig. 35 and τ of 0.13 with no sharpening for Fig. 39.

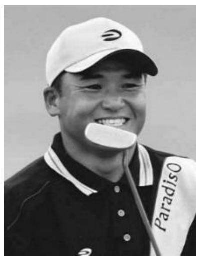
Fig. 2 Original image -1

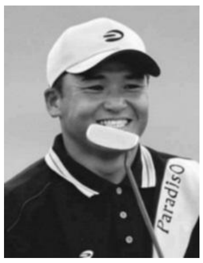
Fig. 3 Preprocessed image

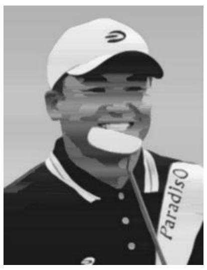
Fig. 4 Cumulative mean image XI_n

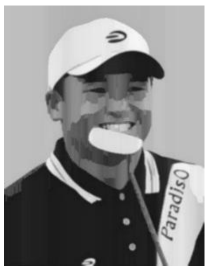
Fig. 5 Cumulative mean image YI_n

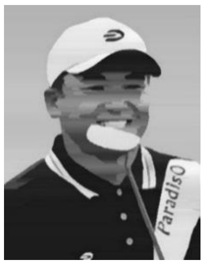
Fig. 6 Transformed image XI_t

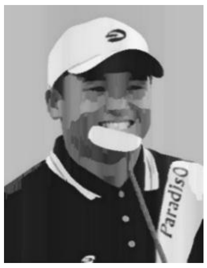
Fig. 7 Transformed image YI_t

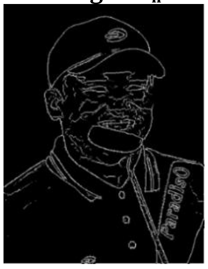
Fig. 8 Binarized image I_b

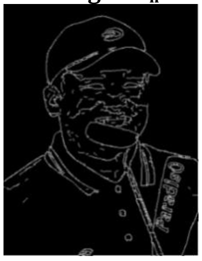
Fig. 9 Pattern corrected and noise eliminated edge image I_E

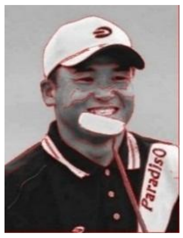
Fig. 10 Detected edges superimposed on original image

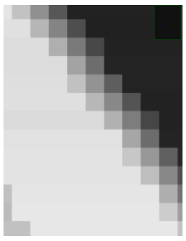
Fig. 11 Edge Ramps at pixel level

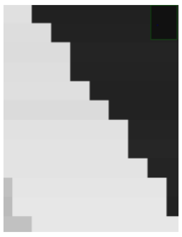
Fig. 12 Edge Ramps converted to step

Fig. 13 Edge Ramps with ztc at pixel level

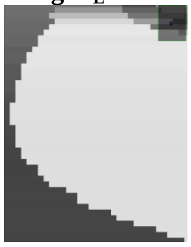
Fig. 14 Edge ramps with ztc converted to step

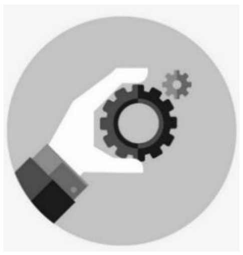
Fig. 15 Original image -2

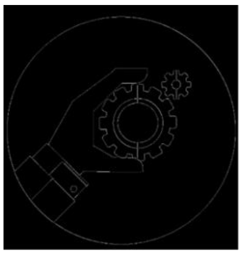
Fig. 16 Binary Edge image I_E

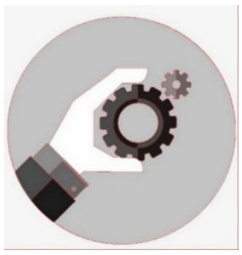
Fig. 17 Detected edges superimposed on original image

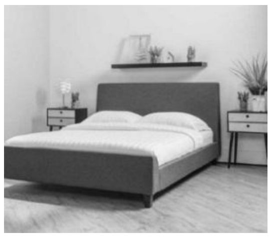
Fig. 18 Original image -3

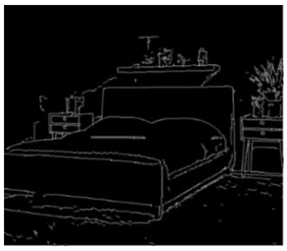
Fig. 19 Binary Edge image I_E

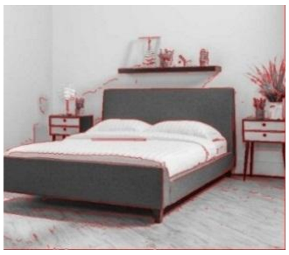
Fig. 20 Detected edges superimposed on original image

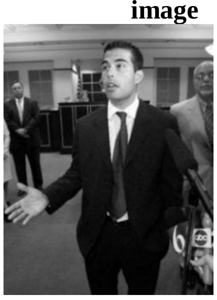
Fig. 21 Original image -4

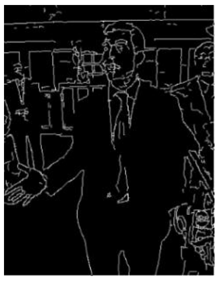
Fig. 22 Binary Edge image I_E

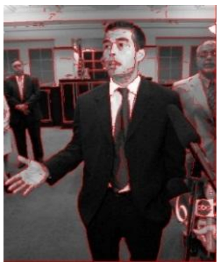
Fig. 23 Detected edges superimposed on original image

TABLE 2 Time taken at various stages and in total

1	2	3	4	5	6	7	8	9	10
Fig.ure	**Dimension**	**No. of Pixels**	**XI_n&YI_n (in s)**	**XI_t&YI_t (in s)**	**I_b (in s)**	**I_{pc} (in s)**	**I_N (in s)**	**Total Time (in s)**	**Time per lakh pixels (in s)**
Fig. 2	450x348	156600	0.322	0.318	0.156	0.355	0.586	1.737	1.109
Fig. 15	859x820	704380	1.182	1.22	0.715	0.962	1.386	5.465	0.776
Fig. 18	512x590	302080	0.515	0.556	0.309	0.636	0.964	2.98	0.986
Fig. 21	450x347	156150	0.336	0.331	0.162	0.429	0.705	1.963	1.257
Average time per lakh pixels: 1.032 s, **Standard Deviation:** 0.204									

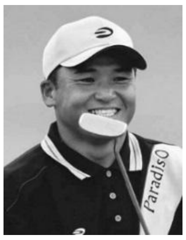
Fig. 24. Original image -1

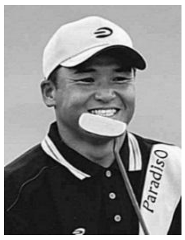
Fig. 25. Preprocessed image with sharpening applied

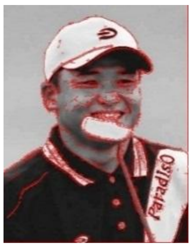
Fig. 26. Processed with τ = 0.10

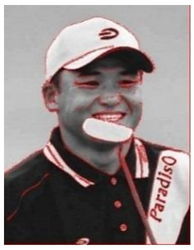
Fig. 27. Processed with τ = 0.20

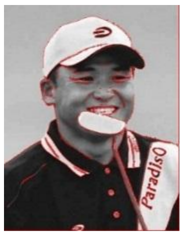
Fig. 28. Processed with τ = 0.25

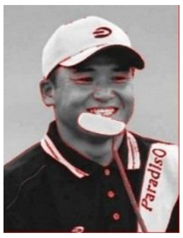
Fig. 29. Processed with τ = 0.30

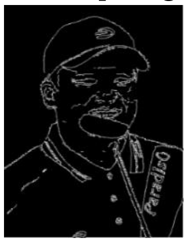
Fig. 30. Edge image when τ = 0.20

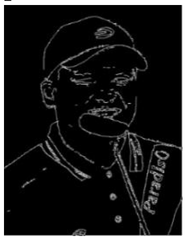
Fig. 31. Edge image when τ = 0.25

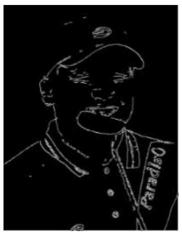
Fig. 32. Edge image when τ = 0.30

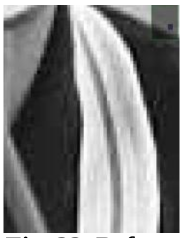
Fig. 33. Before sharpening

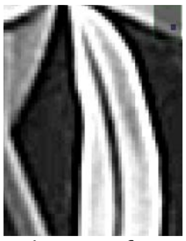
Fig. 34. After sharpening

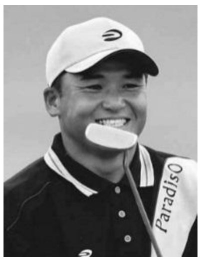
Fig. 35 Original image -1

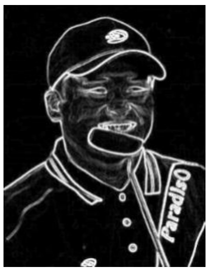
Fig. 36 Using Sobel operator

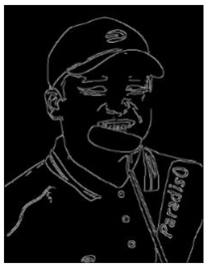
Fig. 37 Using Canny algorithm

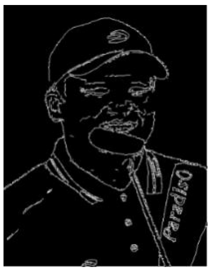
Fig. 38 Using CuMIDT algorithm

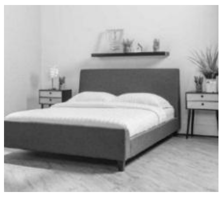
Fig. 39 Original image -3

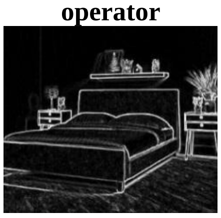
Fig. 40 Using Sobel operator

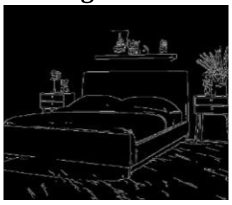
Fig. 41 Using Canny algorithm

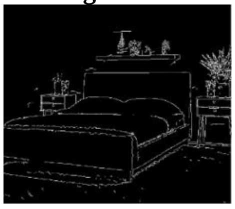
Fig. 42 Using CuMIDT algorithm

V. CONCLUSION

CuMIDT algorithm gives better detection results to FDDB dataset and real dataset, when compared with standard sobel's operator and canny edge detection algorithm. As observed in certain results, some lines may not be representing sharp edge. These are the result of deviation of the pixel intensities from the cumulative mean at the local region. By developing weighted transition points and retaining points with more weights, we can have only strong edges being detected. By applying Gaussian function to smooth out the transition curve, more clear approximation of the line representing the edges can be achieved.

REFERENCES

1. R. C. Gonzalez, R. E. Woods, Digital Image Processing, 3rd edition, Pearson Education.
2. J Canny, “A computational approach to edge detection.” IEEE Trans. PAMI (1986), 8, n ~ 6, pp. 679-698.
3. Qiucheng Sun, YueqianHou, Qingchang Tan, A subpixel edge detection method based on an arctangent edge model, Optik 127 (2016) 5702–5710
4. AssmaAzeroual, Karim Afdel, Fast Image Edge Detection based on Faber Schauder Wavelet and Otsu Threshold, Heliyon 3 (2017), https://doi.org/10.1016/j.heliyon.2017.e00485
5. EserSert, DeryaAvci, A new edge detection approach via neutrosophy based on maximum norm entropy, Expert Systems With Applications 115 (2019) 499–511
6. Shaohu Peng et al., Subpixel Edge Detection Based on Edge Gradient Directional Interpolation and Zernike Moment, 2018 International Conference on Computer Science and Software Engineering, 2018
7. Le Wang, Li Zou, Shengmei Zhao, Edge detection based on subpixel-speckle-shifting ghost imaging, Optics Communications 407 (2018) 181–185
8. Changbao Wen et al., Edge detection with feature re-extraction deep convolutional neural network, J. Vis. Commun. Image R. 57 (2018) 84–90
9. Xiaowei Hu, Yun Liu, Kai Wang, Bo Ren, Learning Hybrid Convolutional Features for Edge Detection, Accepted Manuscript in Neurocomputing, 2018
10. Lucia Romani, Milvia Rossini, Daniela Schenone, Edge detection methods based on RBF interpolation, Accepted Manuscript in Journal of Computational and Applied Mathematics, 2018
11. Luyang Wang, Yuan Shen, Houde Liu, ZhenhuaGuo, An accurate and efficient multi-category edge detection method, Cognitive Systems Research 58 (2019) 160–172

AUTHORS PROFILE

Ganesh Pai, completed his Bachelor’s degree in the year 2006 and Master’s degree in 2009 from NMAMIT, Nitte, Karkala. He has guided several UG and PG projects and published several papers in National and International Conferences/Journals. His area of interest includes Application Programming, Web Technologies and Image Processing.Currently he is pursuing his Research work in Digital Image Processing.

M SharmilaKumari, obtained her Ph.D. in the field of Image Processing from Mangalore University, Mangalore, Karnataka, India in the year 2012. Currently she is heading the Computer Science and Engineering department at P A College of Engineering, Mangalore, Karnataka, India. Her areas of specialization include Pattern Recognition, Image Processing, Embedded Systems and Microprocessors. She has published around 80 research articles in International and National Journals and Conferences.She has successfully completed three collaborative research projects with Moscow State University, Moscow, Russia awarded under DST-RFBR sponsorship and also completed the bilateral workshop on Emerging Applications of Computer Vision-2011 with Moscow State University, awarded under DST-RFBR sponsorship.