An Efficient Contour Based Fine-Grained Algorithm for Multi Category Object Detection

Rafflesia Khan, Tarannum Fariha Raisa, and Rameswar Debnath
Computer Science and Engineering Discipline, Khulna University, Khulna, Bangladesh
Email: {rafflesiakhan.nw, fariaraisa2192}@gmail.com, rdebnath@cseku.ac.bd

***Abstract*—Object detection from a real-time image is gaining increasing research interest in recent years as it can substantially facilitate a wide range of content-based multimedia applications. Detecting objects from real time image with fine grained details requires extensive amount of preprocessing, computation and time. In addition, multi category object detection is a very complex and diverse problem domain. Some traditional object detection and recognition models previously train their model with huge number of rich and highly annotated images and then divide an input image into set of bounding boxes and calculate the confidence score for each object category in the image. Most of the existing approaches require huge amount of time and computation for object detection. On the other hand, some models only work with local images where image has a single, focused and center-based object. In order to overcome such limitations of existing models, we are proposing a Region of Interest (ROI) based object detection model. In our proposed work, instead of using a fixed number of bounding boxes (e.g. n×n) or working with local images, our model identifies all the ROI at any location of the image. We have tested our algorithm on a number of benchmark datasets for fine-grained object detection. In our work, we have demonstrated significant efficiency as well as accuracy in term of object detection. We have significantly improved the algorithm to detect object irrespective of location, number of object in the image, object overlapping and minuscule objects. Our proposed model can reduce the noise in an image, and thereby, can identify ROI even from poor quality image, noisy background, irrelevant context and misleading feature. To demonstrate the accuracy of our proposed algorithm we have introduced a feature matching approach to identify the detected objects correctly.**

***Index Terms*—artificial intelligence, object detection, region of interest, blur image, laplacian variance, OpenCV**

I. Introduction

In the era of Artificial Intelligence (AI) and Robotics, object detection is one of the fundamental functionalities. Machines need to detect objects or ROI from an image with limited pre-processing, computation and time without compromising accuracy. There are several challenges in object detection problem domain. For example, sometimes the difference between the background and foreground of an image is very limited. In Fig. 1(a) the bird named brown creeper has almost the same texture as the tree bark and in Fig. 1(b), the fruit hanging among lots of leaves is hard to identify. Another challenge is that, real life images are taken with very noisy background which can totally disrupt any object detection algorithm and can point out a completely wrong target object. Therefore, object detection algorithms require huge computation and more complex cluster definition with more detailed features of an object to overcome such challenges.

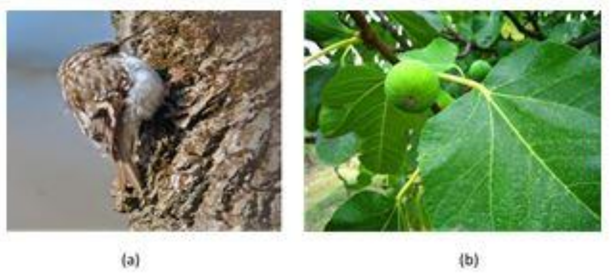


Figure 1. Sample images that can be hard for object detection

Most of the known neural network based model such as FAST-RCNN [1], YOLO [2], RCNN [3], have used a big data set and high-performance computers to get an accurate result with satisfactory time. These models are dividing image in to smaller boxes and then scanning an image pixel by pixel using bounding box and searching whether there exists an object or not. For these operations, they need huge time and computation as well as energy.

On the other hand, some of the traditional object detection approaches like [4], [5] and [6] work with single object images only, therefore, they cannot be used for object detection from global image or image with more than one object. Another current approach of object detection is to choose a classifier for the target object and evaluate it at various locations in a test image and finally run a trial and error approach to match the region with the provided classifier. For example, Systems like Deformable Parts Models (DPM) [6] use a sliding window where the classifier is running at evenly spaced locations over the entire image. This is very time consuming and the accuracy of the object detection depends extensively on how the classifier has been designed. In addition to these existing limitations of time and performance, multi category object detection [7] and [8] is even more complex and therefore, this is a growing research field. Let's consider some sample images, depicted in Fig. 2 in order to understand the challenges of object detection problem domain.

Manuscript received July 28, 2018; revised December 6, 2018.

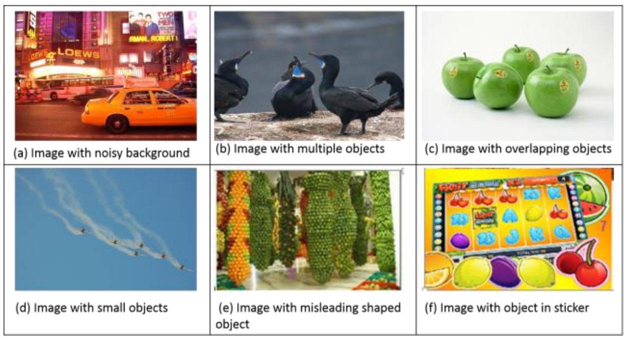

(a) Image with noisy background
(b) Image with multiple objects
(c) Image with overlapping objects
(d) Image with small objects
(e) Image with misleading shaped object
(f) Image with object in sticker

Figure 2. Images with different kinds of object detection and recognition challenges.

- Most of the real-life images have very noisy background with several objects like Fig. 2(a). Object detection from these kinds of images is very complex as differentiating the foreground objects from background requires extra effort and computation.
- Not all objects in an image exists at the center of an image as salient ones [4], [6] like Fig. 2(b). Traditional object detection approaches [4], [5] and [6] work with single object images. But most of the real time images contain multiple objects at any location of the image with different focus length.
- Traditional models usually ignore the images with overlapping objects like Fig. 2(c). But overlapping objects need to be detected for recognizing each object from an image separately.
- Object detection model like YOLO [2] struggles with small objects that appear in groups, such as flocks of birds or airplanes like Fig. 2(d). However, we need a detection model which is capable of detecting every kind of objects.
- To recognize the correct context, only detecting shape is not enough. For example, in Fig. 2(e) the shape of a connected contour would mislead the target object. Therefore, only shape and color are not enough to identify any object.
- Real time object and sticker or emoji need to be differentiated from each other, in Fig. 2(f). Therefore, the proposed algorithm also needs to consider the 3d effect from the image. When we are trying to perform machine learning for a smart- refrigerator to identify tomato and apple correctly, we also need to make sure that it can differentiate 3d object and 2d image.

Therefore, our goal is to define a fine-grained algorithm that can perfectly detect objects addressing all the above-mentioned complexities. In our proposed work, we have addressed several open challenges of object detection. The main contributions of our paper are as following

- We have defined a fine-grained algorithm to detect all the objects or ROI on the foreground of an image and create cropped single object images for each of them. Therefore, instead of dividing the image in nxn bounding boxes, this algorithm has reduced the number of bounding boxes to n, where n is number of region of interest in the image. This has reduced the computation time and complexities drastically.
- Our algorithm can detect objects from images with multiple objects at any location. Poor quality or even blur focused object in image and overlapped objects in the image do not compromise the accuracy in our proposed model.

Finally, we have evaluated the performance of the proposed algorithm against benchmark data as well as against existing systems.

II. Proposed Fine Grained Algorithm for Object Detction

Our proposed work mainly works for identifying every single object from an image regardless of object's location, poor quality and object overlapping. Each image can be represented as a collection of region of interests (ROI) or objects as in equation (1). Here ROI is an area that contains a connected contour or object.

$$IMG = \{ROI_1, ROI_2 \dots \dots \dots \dots, ROI_n\} \quad (1)$$

These ROIs are used for object detection process. This proposed algorithm requires only the contour that contains the object for computation and analysis, rather than 300 × 300 bounding boxes for each image [1], [2]. Therefore, in any image if there are 10 objects it would have 10 boxes or contour instead of 300-500. This has significantly improved the performance and accuracy as well. In this section, we are going to explain our object detection algorithm. In order to verify the accuracy of our algorithm, we have also introduced an object identification technique. In our object identification process, each of the objects or detected ROIs has been represented as a vector of shape (S), texture (T) and color (C) as shown in equation (2).

$$ROI = V < S, T, C > \quad (2)$$

In case of several research works [9], the images have been resized to 100 × 100 to minimize the computation, however resizing a large image into such smaller size often causes loss of context and therefore the accuracy of object detection gets sacrificed. Therefore, we have picked the minimum size of each image as 700 × 500. This size, 700 × 500, gives us better context and rich feature which ultimately improves the accuracy and we also find more contextual information ultimately reduces the time required for object detection. At the same time, it did not cost us too much time, therefore, larger image size doesn't act like an obstacle for our work. After resizing the image, we have followed the following approach. The first step of our proposed work is to determine the region of interest. In this section, we have described how to determine all the ROIs from an image. For that we have defined the ROI along with their shape, color and texture feature which have been used by our algorithm to recognize the corresponding object contained in each ROI. In this section, we have described our algorithm in detail as following

A. Blurriness Detection

After resizing the image, the first step is to detect if the image is blurry or not. Blurry images are captured by focusing on target objects and ignoring the background, therefore, they are easy to process and suitable for immediate edge detection as well as other kinds of image processing. An example is shown in Fig. 3(b). On the other hand, a non-blurry image has rich background where target objects are not focused, as shown in Fig. 3(a). Non-blurry images need extra effort on differentiating the background from foreground and making the foreground objects much focused. Otherwise, the edge detection for these non-blurry images will give inaccurate result.

For blurry and non-blurry image, we have followed the survey work described by Szegedy, *et al.* [10]. Among 36 different methods to estimate the focus measure of an image and compute the 'blurriness metric' we have used variation of the Laplacian by Pech-Pacheco *et al.* [11]. We take a single channel of an image (grayscale) and convolve it with a 3 x 3 Laplacian kernels as shown in (3).

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{3}$$

Kernel: The Laplacian kernel.

If the Laplacian variance (Lv) (i.e. standard deviation squared) of the response falls below a pre-defined threshold (Lv < 300), then the corresponding image is considered as a blurry image; otherwise, it is a non-blurry image. In our work, we have used different methods for processing blurry and non-blurry images which has minimized the computation time significantly. The Lv of the image shown in Fig. 3(a), is 1176.83 which is greater than threshold (300), therefore, our algorithm identifies it as a non-blurry image. Later in this paper we have shown the output for object detection for this non-blurry image shown in Fig. 3(a). On the other hand, the Lv of the image shown in Fig. 3(b) is 155.24. Therefore, it is comparatively a blurry image. Later in this paper we have also shown the output of object detection using our proposed algorithm for blurry image shown in Fig. 3(b).

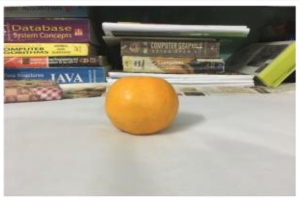
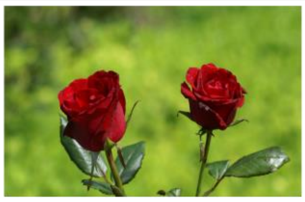
	
(a) Non Blurry image: High Lv variance image where Object is not focused and have rich background .	(b) Blurry image: Low Lv variance image where Object is focused and have blurry simple background .

Figure 3. Blurry and non-blurry image

B. Edge Detection

Once the image is identified either as blur or non-blur, the next step is to detect edges of each objects or ROI. This is not a straight forward process. We have processed blurry and non-blurry image separately for edge detection.

1) Processing blurry image

Instead of following the traditional way to use local images with single object for object detection, we have targeted to detect all the connected contours of an image where each contour may contain single or overlapped objects. The first step to achieve this goal is to detect the edge of each contour of the image. There are several well-known approaches to detect edges from image which include Sobel, Canny, Prewitt, Roberts, and fuzzy logic methods. However, we observed that applying any of these algorithms to local images with unsharpened edge as shown in Fig. 4(b) or a global image having multiple objects with colorful background Fig. 4(c) really gave poor results. Which is not efficient.

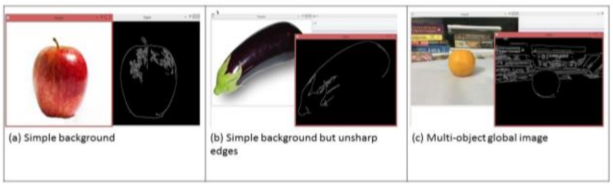

(a) Simple background
(b) Simple background but unsharp edges
(c) Multi-object global image

Figure 4. Defected results of edge detection using traditional method directly.

Most of the recent works on object detection [4], [5] use time consuming, complex and previously developed algorithms to differentiate foreground from background but the output is not quite satisfactory. In order to overcome this limitation, we have developed a very easy and less time-consuming algorithm to differentiate foreground from background more efficiently. In addition, edge detection is susceptible to noise. Therefore, we have removed the noise from the image using unsharp masking [12] with a 9x9 Gaussian filter. Here we have used this gaussian smoothing filter and subtracted the smoothed version from the original image (in a weighted way so the values of a constant area remain constant). For blurry image, we sequentially sharpen the edges of every contours in the image and detected edges by applying Sobel operator [13]. Sobel operator computes an approximation of the gradient of an image intensity function. And it also combines Gaussian smoothing and differentiation. If the image to be operated is A. We have calculated two derivatives both horizontal and vertical as shown in equation (4) and equation (5):

a. Horizontal changes: This is computed by convolving A with a kernel Gx with odd size. For example, for a kernel Gx size of 3, would be computed as:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * A \tag{4}$$

b. Vertical changes: This is computed by convolving A with a kernel Gy with odd size. For example, for a kernel Gy size of 3, would be computed as:

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * A \tag{5}$$

Now at each point of the image we calculate an approximation of the gradient (G) in that point by combining both Gx and Gy: This result makes the edges of a blurry image perfectly sharp for edge detection. The approximated gradient which highlights the edges of each contour.

$$G = \sqrt{G_x^2 + G_y^2} \qquad (6)$$

Then we generate saturated version of the detected edge to minimize the amount of computation and sharp the edge for each contour. Once all the edges are sharpened, we get better result for recognizing each contour from the input image (IMG).

The second step is to extract the colorfulness of the objects according to its brightness by using the saturation information. To get the saturation information of the image, we split the magnitude of the image obtained in the first step. Thirdly, the input image is converted to binary. This makes it easier to extract each contour containing potential target objects in the image. To binarize the image, we have used Otsu Thresholding [14]. This method treats the image as a bimodal image. It finds the histogram peaks that best describes the image and selects a value that lies between the two peaks [14]. Then we have constructed and applied a closing kernel to 'close' gaps between 'white' pixels. To get rid of the few areas with small patches, we have used median filter which is best known for removing noise from image. It calculates the median value in the neighborhood area and assigns that value to the corresponding pixel of image. This application of median filter also smoothens the edges of each contour. All these above preprocessing on blurry image finally makes the image perfectly ready for contour detection. Fig. 5 shows edge detection result for the blur image shown in Fig. 3(b).

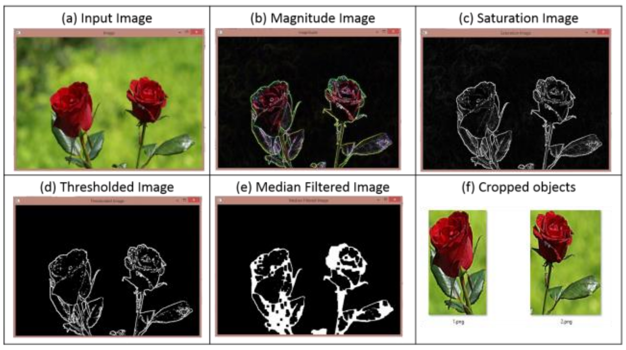


Figure 5. Object detection result for blurry image.

2) *Processing non-blurry image*

Edge detection process in non-blurry image is initially a complex work as a non-blurry image does not have focused objects and mostly has rich background. Our first concern was to differentiate image foreground from its background and make the foreground objects much focused. The other concern is that, the edges are already sharp, but not enough. For better performance, we have processed the images with following procedure.

At first, we have converted the image from RGB color space to HSV Cylindrical coordinates as the HSV color model is often preferred over the RGB model. The HSV model describes colors similarly as the human eye tends to perceive color. The HSV values are often used to determine the location of a specific object or color for object searching models [15]. Therefore, converting the input non-blurry image to HSV channel, highlights all the objects in the image that is much visible to human eyes. The HSV conversion of input image shown in Fig. 6(a) is shown in Fig. 6(b) which clearly shows that converting the input image in HSV channel makes the foreground objects much focused which is beneficial to further edge detection process.

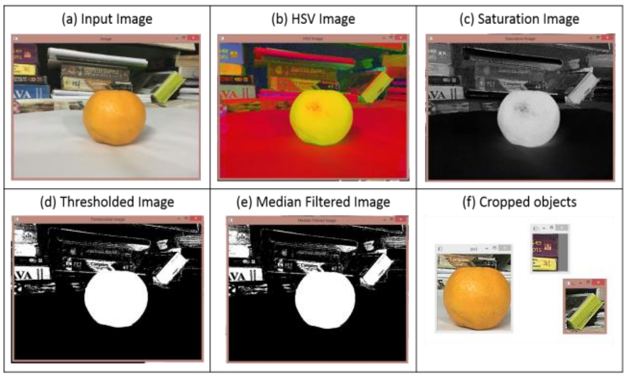


Figure 6. Object detection result for non-blurry image.

Now we split the output image from the previous step Fig. 6(b) and take only the saturated part of the image Fig. 6(c). We binarize the saturated image and apply a closing kernel to 'close' gaps between 'white' pixels. Finally, median filter is applied as shown in Fig. 6(e) for removing noise. These processes sequentially make target objects more focused and sharpen edges for smoother connected component detection. Fig. 6 shows the output for object detection for this non-blurry image previously shown in Fig. 3(a).

C. *Contour Detection*

Contours can be explained simply as a curve joining all the continuous points (along the boundary) having same color or intensity. Finding contours is like finding foreground object from background. The contours are a useful tool for shape analysis and object detection and recognition. After preprocessing blurry and non-blurry image differently as described in previous section, we get a median filtered image which is perfect for contour detection. If we look at the image obtained in the previous step, we see that we have some large white sections in the black area. In our image, we could think of it as the area that is formed by joining the boundary of white pixels. This is one of several ways to find blobs (binary large objects) in an image using OpenCV [16].

After detecting all the connected contours of the image, we simply crop each connected contour. In other words, we cut the global image into some small local images containing only one object per image. One of the major challenges of contour detection is that it needs to separate irrelevant object from the target object. In order to overcome the challenge, we considered only those objects those cover more than 8% area of the actual image. Fig. 5

and Fig. 6 show the consecutive 6 steps to determine the different ROI or objects from an input image. In the rest of this paper, we would mention these generated images contain single couture as ROI image. At the final step, we have a set of images (ROI), each of which contains one single connected contour. Each of these contours do not always contain a single object. They may have overlapping multiple objects. By analyzing each of these connected contours separately in parallel we have improved the performance and accuracy of object detection significantly in our work.

D. Manage Objects Overlapping

Objects can exist overlapping one another or closely connected to each another in a single connected contour obtained in previous section (2.3). Overlapping of different kind of objects need to be handled for identifying those objects separately. Moreover, overlapping of same kind of objects need to be handled for knowing their number of instances. In our object detection model, we have been able to separate two or more overlapped objects. For doing this, we mainly focused on sharping the overlapped object's edges between two or more objects so that contours of different objects can be identified individually. For sharping the overlapping edge, we have analyzed stylization filter, detail enhance filter, edge preserving normalized convolution filter and Pencil sketch color filter, details explained in [17]. In addition, for handling overlapping of some different kinds of image we have applied different kinds of thresholding which includes, binary threshold, inverse binary threshold, truncate threshold, to zero threshold, to zero inverse threshold, details explained in [18]. One specific filter doesn't fit for every kind of image. However, among all these filters we have achieved satisfactory result for stylization filter and to zero inverse threshold. For every overlapped contour, we apply these filters and then run the edge detection and contour detection approach again to create separate ROI image. Fig. 7 shows that our model can separately identify every overlapped object from an image.

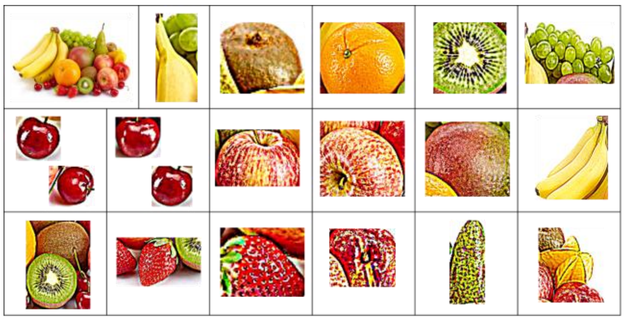


Figure 7. Object detection with overlapping objects.

E. Eliminating Misleading Contour

In our work, we are focusing on detecting edges of objects so that we can detect the contour or boundary shape of an object and crop that as a single image from main image. Besides handling multiple objects or overlapped objects another challenge is that objects can have so many edges on their surface which can mislead the actual contour edge of the object. This misleading edge can result in a complete separate contour. For example, objects like fruit, flower etc. can have foreign object e.g. water drop or any other spot on their surface that can create new edges or contour which can mislead the contour detection. Fig. 8(a) shows a pear having water drops on its surface and Fig. 8(b) shows a rose with many edges of its petals which can mislead the contour detection. Our model is also able to ignore these misleading contours or edges very efficiently. We have used closing kernel to 'close' gaps between 'white' pixels which helps to eliminate the small edges from surface of object. In addition, converting the image into HSV version make every less visible edge e.g. water drop to fade away. For every output images (ROIi) from the set ROI as in equation (1), we simply detect the largest ROI. We ignore all relatively smaller contour those are less than the threshold value (e.g. 8% of complete area). Therefore, we can solve the problem by ignoring any foreign object which could mislead the computation result.

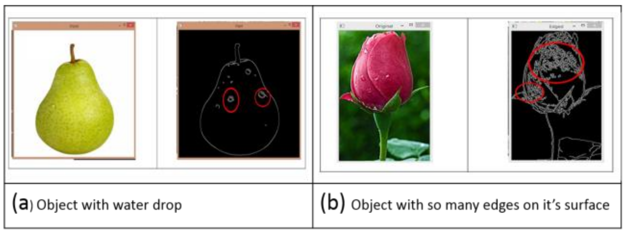

(a) Object with water drop
(b) Object with so many edges on it's surface

Figure 8. Images with misleading contours and unnecessary edges.

At this stage, we get a set of ROI image from the input image (IMG). Each ROI contains exactly one object. Each object in each ROI image has it's own shape, color and texture features. These features can be used for identifying corresponding object in each ROI. Once we have the set of objects from the image, we can further use these detected ROI or objects for different application. Some of the application of the output of our proposed algorithm are object recognition by feature matching, object recognition by deep learning and so on.

III. Feature Matching for Object Recognition

Once all the ROI or contour of the input image (IMG) are detected, the next step is to identify the object contained in each ROI. In this work, we have described a feature matching technique for object identification. We have detected shape, color and texture features for this purpose. This section describes the feature detection and matching process.

A. Shape Detection

For shape detection, the first step is to take a black mask having same size as the ROI image (ROIi) and draw the contour on it. Bitwise adding this mask on ROI image gives an image with only target object on a black background as shown in Fig. 9. Finally, we replace the black background with a white one. Therefore, at the end of this step, we have the shape of the object in a local image (ROIi) with a white background. This procedure helps us to overcome the problem where sometimes background could be in the same range of the threshold for connected contour as like the bird in Fig. 1(a) and can cause additional metadata about the object which ultimately can mislead the shape detection process. The next step is to get more information e.g. color, texture about the detected connected region in each ROI (ROIi).

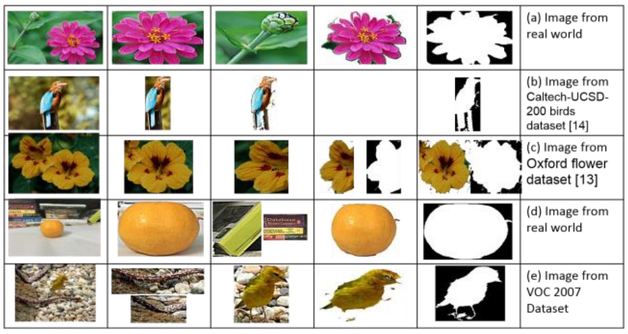

(a) Image from real world
(b) Image from Caltech-UCSD-200 birds dataset [14]
(c) Image from Oxford flower dataset [13]
(d) Image from real world
(e) Image from VOC 2007 Dataset

Figure 9. Outputs of shape detection of objects.

B. Color Detection

Color is a feature that can capture a certain visual property of an object. Histograms are useful because they are relatively insensitive to position and orientation changes and they are sufficiently accurate. However, they do not capture spatial relationship of colored regions, therefore, they can sacrifice the accuracy instead of improving it. Therefore, we have chosen color moments as color feature instead of traditional histogram. Models like [19], [20] have use three color moments e.g. Mean in equation (7), Standard deviation in equation (8) and Skewness in equation (9) of an image's color distribution. To calculate the feature vector, we first divide each RGB image into its component channels. For each channel, we calculate these three moments making a total of 9 features for each image. The three-color moments are defined as the first-order (mean), the second-order (standard deviation), and the third-order (skewness) color moments. These color moments have been proved to be efficient and effective in representing color distributions of images.

- Mean (First order Moment):

$$E_i = \sum_{N}^{j=1} \frac{1}{N} p_{ij} \tag{7}$$

- Standard Deviation (Second Order Moment):

$$\sigma_i = \sqrt{\left(\frac{1}{N} \sum_{N}^{j=1} \left(p_{ij} - E_i\right)^2\right)} \tag{8}$$

- Skewness (Third order Moment):

$$s_i = \sqrt[3]{\left(\frac{1}{N} \sum_{N}^{j=1} \left(p_{ij} - E_i\right)^3\right)} \tag{9}$$

here the value of ith color channel at the jth image pixel is pij.

C. Texture Detection

We have used uniform Local Binary Pattern (LBP) [21] for texture detection. For each of the ROI (ROIi) we calculate the normalized LBP histograms within a range of 0 to 255. As LBP only supports grayscale image, each ROIi is converted to grayscale image. This LBP mask is computed by configuring the radius of the neighborhood equal to 3 and the number of points = 24. Once we have the mask, we simply calculate the LBP version of ROIi, then we calculate the LBP histogram and normalize it. Next, we append the normalized histogram to a list as the texture feature of each object or ROI.

D. Feature Matching and Object Identification

In this work, we have used feature matching technique for object identification. For this we compare the each ROI image with previously stored dataset images and detect corresponding object of the actual input image. For recognizing an object, we compare the shape, color and texture feature of each ROI$_i$ identified in previous steps with corresponding dataset image feature. For feature mapping, we have used several distance measure methods e.g. Correlation, Intersection, Hellinger, Chi-square, Euclidean and City block. After applying all these different methods on several benchmark image datasets. e.g. VOC 2007. ImageNet, we have come to the point that Chi-square and City block distance measure techniques are giving better result for feature matching in our model.

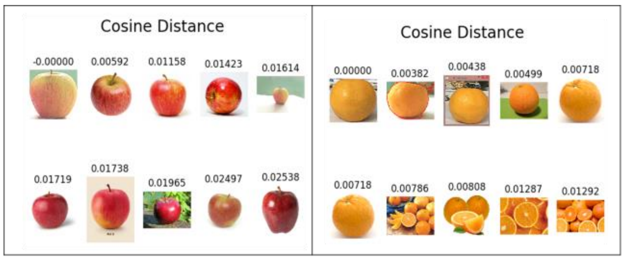


Figure 10. Feature matching results for an apple and an orange image as input.

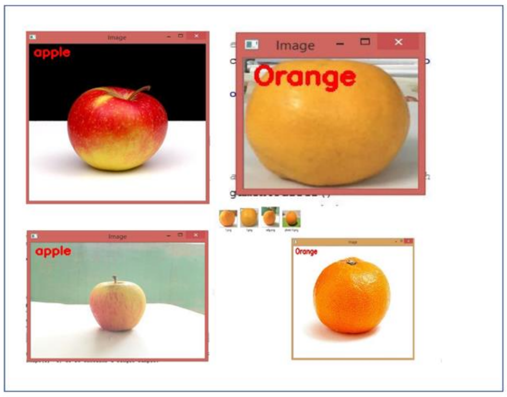


Figure 11. Feature matching recognition results for different image as input.

For a ROI image, the feature matching technique generate distance for each image in the database. For each object recognition, we choose top 10 objects with highest match and lowest distance. Finally, we consider our input object belongs to the class of those 10 matched images. Fig. 10 shows some feature matching results. The cosine distance for the target object image is 0.000 and Fig. 11 shows the object detection result using feature matching.

Therefore, in a brief the complete process of our proposed work can be described as the flowchart shown at Fig. 12. Each of the images are preprocessed, such as larger images are resized to 700×500, background are removed. On the very first step, based on Laplacian variance, the input image is detected as either blur or non-blur image. Once the preprocessing is complete, different methods are used for edge detection described in section 2.2 for blur and non-blur image. After the edge detection preprocessing is complete, all existing contour or ROI are detected on next step as described on 2.3 section. The next step is to determine if there are any overlapped objects in any of the deleted contour as described in 2.4 section. If there is overlapped contour, they are separated and extract as separate ROI. If any ROI has small foreign objects or misleading edges, in the next step. Then next step is to crop all the detected ROI as images. Now we have n images from the main image, where n is the number of ROI or object in the main input image (IMG) as like equation (1). Our next steps are applied on each of the ROI in parallel. The first step is to determine the shape of the corresponding ROI as described in section 3.1. Only shape detection is not enough. Therefore, in the next step, we detect the color as described in section 3.2 and texture as described in 3.3 section. Finally, each ROI is matched with the database images and the matched result shown in Fig. 10, Fig. 11 is quite satisfactory. We have also used SVM to test the accuracy of the object detection for each ROI and the result is shown in Fig. 13. For the ImageNet, VOC 2007 as well as real time image captured by us, the detected objects using our proposed model has been identified using SVM with 100% accuracy.

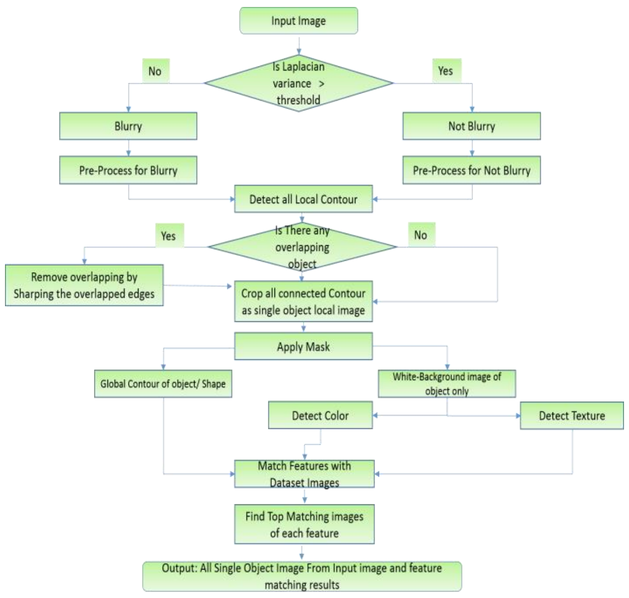

Input Image
Is Laplacian variance > threshold
No
Yes
Blurry
Not Blurry
Pre-Process for Blurry
Pre-Process for Not Blurry
Detect all Local Contour
Is There any overlapping object
Yes
No
Remove overlapping by Sharping the overlapped edges
Crop all connected Contour as single object local image
Apply Mask
Global Contour of object/ Shape
White-Background image of object only
Detect Color
Detect Texture
Match Features with Dataset Images
Find Top Matching images of each feature
Output: All Single Object Image From Input image and feature matching results

Figure 12. Flow chart for the proposed fine-grained object detection and identification algorithm.

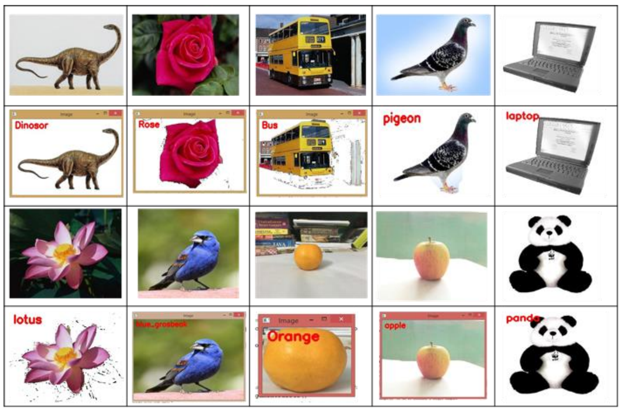


Figure 13. Image classification or mapping result of SVM exploiting feature and object detected by proposed model

IV. Performance Evaluation & Comparetive Analysis

A. *Performance and Accuracy*

We have applied our algorithm on the reference image used in [9] to compare the accuracy and performance. The proposed approach significantly outperformed the existing [9] approach by successfully detecting object in both local and global images. The object detection method described in [9], requires 100 - 200 bounding boxes to be computed, even after removing the background and ignoring the blur objects. But, in our case, we just have detected the connected region of interest and detected the bird from that image faster than existing approach. Fig. 14(b) show the output of [9] and Fig. 14(d) shows the output of our proposed algorithm. Instead of creating a number of bounding boxes, we just have two ROI where one of them contain the target object, in this case the bird.

Fig. 14(a) shows the reference image from [9] with simple background and it contains a bird on a branch. According to our algorithm this is a blurry image. Fig. 14(a) has two objects. Instead of having a time-consuming computation for 100 - 200 bounding boxes and determining the confidence for each box as in Fig. 14(b), we have detected the region of object in the image, so we get 2 objects for this image as shown in Fig. 14(c), finally we have removed the background of image and obtain ROI containing only the target object Fig. 14(d) that we want for further analysis and object identification.

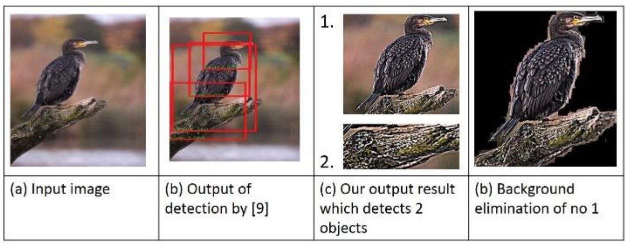

1.
2.
(a) Input image
(b) Output of detection by [9]
(c) Our output result which detects 2 objects
(b) Background elimination of no 1

Figure 14. Comparative results of object detection.

In our work, we have concentrated on the connected region and detached all the ROI from images. We also have used a threshold value to filter minuscule connected region with area less than 8% of total image to ignore the irrelevant contours. Therefore, we have overcome one of the limitations of [4] which can't detect small objects and overlapped connected objects. Our proposed algorithm can successfully detect any smaller area larger than the threshold area. Fig. 15 shows the performance of our proposed algorithm on 2 images with multiple minuscule objects from VOC 2007 challenges. In the image with birds, all the birds are detected from a group of birds and from the image with airplanes, all small airplanes are also separately detected using our proposed model.

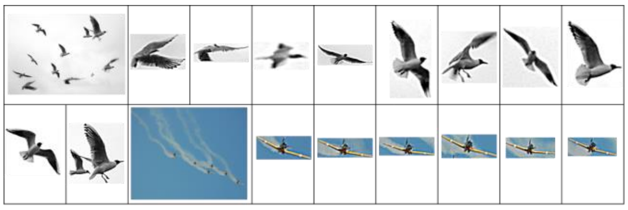


Figure 15. Object detection result from global images with minuscule objects.

Our proposed algorithm has successfully overcome the limitation of salient object detection described in [5] and the huge computation [4] and computing huge number of bounding box [9]. Once we can have detected connected contour or ROI, we already know how many potential objects is there in the image in our work.

B. *Improvement in Noise Detection*

In this proposed work, the background of the image has been removed and the noise has been reduced significantly, which ultimately reduce the amount of computation. Therefore, we will be able to run this application in mobile phone without any cloud-based service. As we process only the contour with target objects, we have achieved a better result by using our very simple and easy technique which requires limited energy as well.

C. *Object Detection from High Texture Background*

In our work, we have successfully detected the changes in texture for any image. Fig. 16 shows that our proposed algorithm can successfully detects objects with skewed texture difference as well as similar shape.

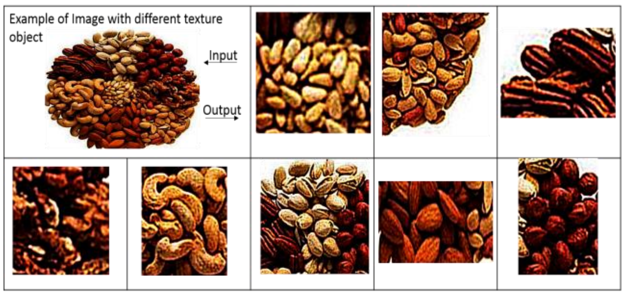


Figure 16. Detected objects of different texture even with same shape.

D. *Medical Image Processing*

Medical image processing is completely different research area, however in case of medical image recognition, it is necessary to detect anomaly from an image with less color and texture variant. Our proposed work can be used for medical image recognition with very faster result with maximum accuracy. Medical robotics often need to detect any un-uniform object such as if a bone is broken or if there is any alien object such as the pin shown in Fig. 17. Our proposed algorithm can successfully identify the alien object those are not matched with the context.

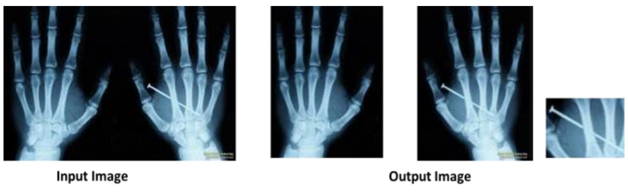


Figure 17. Object detection from an X-ray.

E. *Object Detection from Local and Global Image*

Most of the traditional approaches work on local image containing only a single object, then place that object in the center of the image in order to detect it. In [4], [5] and some other works, it is possible to detect objects only when the object is in the middle of the image and there is only one object in the image. Without using an expensive as well as time consuming deep learning algorithms [1], [2], [3] we have successfully detected objects from image containing more than one object as shown in Fig. 14, Fig. 15, Fig. 16, Fig. 17 and Fig. 18, objects at any location of the image and even overlapped objects as shown in Fig. 7, 16. Therefore, one of our significant contribution is that with limited energy and resources we can detect object from a real-time image, local or global, without any kind of preprocessing. Our proposed work is comparatively fasted than the existing state of art works.

F. *Identify Multiple Objects in Global Image*

As shown in Fig. 18, the global image contains more than one object even with same shape and completely dark background. However, our algorithm has been successfully identified each of the objects with precise accuracy. For this test case, we have run our algorithm on an image from VOC 2007 challenges [22].

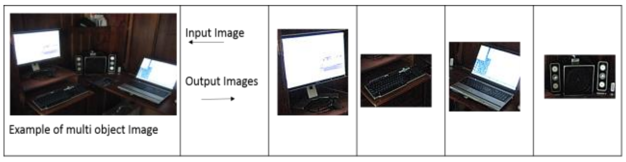


Figure 18. Detection of multiple object with same shape and dark background.

G. *Identify Multiple Objects from any Location of Image*

In most of the existing works, SOD [4], the fine-grained algorithm described by Aniela *et al*. [5] and some other works as [2], [9], [10], [23] the object can be detected if it is at the center of the image and also it is the only focused object of the image while the background is dark. Therefore, in existing works, an object cannot be detected if the following criteria is not satisfied:

- Object is at the center of the image,
- Background is blurred,
- Object at the center is focused.

However, in our proposed work, we have successfully identified each object from an image as shown in Fig. 19 even if the above conditions are not satisfied. Fig. 19 also shows that our algorithm can detect objects at any location of the image. In addition, our algorithm can detect objects from a very noisy image significantly better than other state of art works.

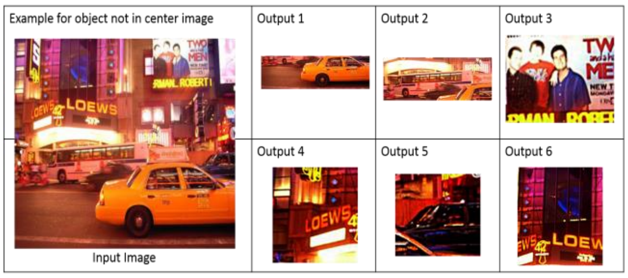


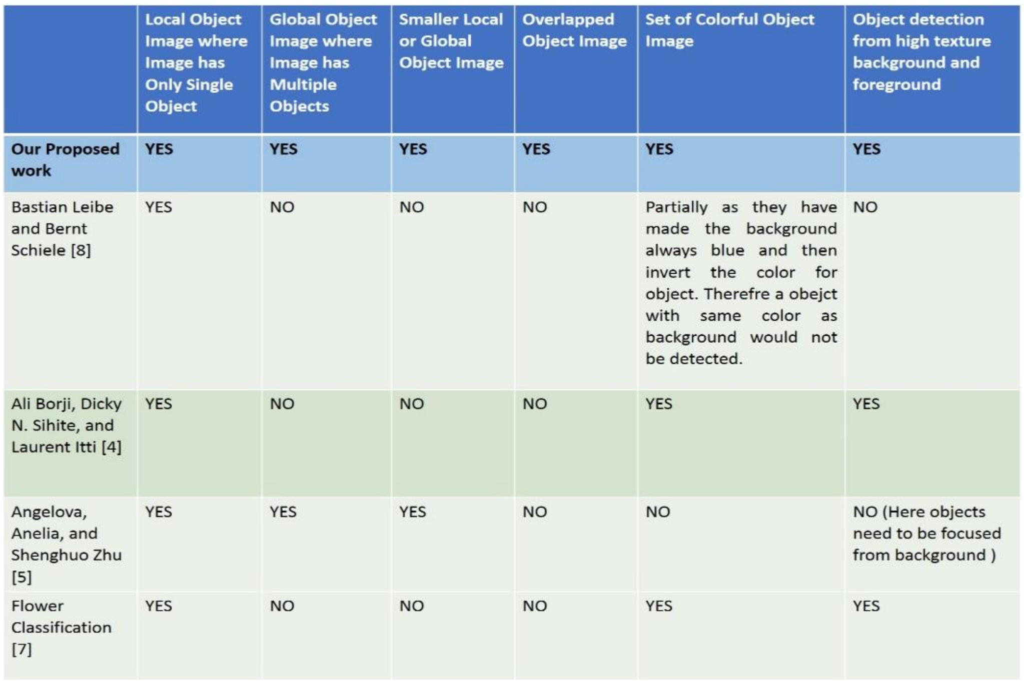
Figure 19. Object detection from any place of the image.

H. *Comparison with Some Model in Object Detection*

We have established a comparative analysis for several well-known research works to demonstrate the comparative performance evaluation of our work.

We have summarized a comparative analysis of our proposed algorithm with other state of art work as in Table I. We have achieved better result for object detection and recognition addressing some crucial challenges for object detection problem domain.

TABLE I. Comparison on Different Aspect of Our Algorithm with [4], [5], [14] and [18]



	Local Object Image where Image has Only Single Object	Global Object Image where Image has Multiple Objects	Smaller Local or Global Object Image	Overlapped Object Image	Set of Colorful Object Image	Object detection from high texture background and foreground
Our Proposed work	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**
Bastian Leibe and Bernt Schiele [8]	YES	NO	NO	NO	Partially as they have made the background always blue and then invert the color for object. Therefre a obejct with same color as background would not be detected.	NO
Ali Borji, Dicky N. Sihite, and Laurent Itti [4]	YES	NO	NO	NO	YES	YES
Angelova, Anelia, and Shenghuo Zhu [5]	YES	YES	YES	NO	NO	NO (Here objects need to be focused from background)
Flower Classification [7]	YES	NO	NO	NO	YES	YES

V. Coculusions

In this paper, we have proposed ROI based multi-category object detection approach. The significant contribution of our object detection algorithm is reducing the huge amount of computation without compromising the accuracy. Our model aims to detect all the connected contours from any kind of image which can ensure higher accuracy while dealing with multi-object images. Another important contribution of this research work is that it can successfully identify objects from poor quality image, image with overlapped object or closely connected objects and noisy images. The proposed work demonstrated better performance in terms of memory, time and computation. Using our algorithm, we were able to ignore misleading objects and detect single, multiple, small, overlapped as well as group of objects from both blurry and non-blurry images. For justifying the performance of our work, we have also developed a feature matching based and an SVM based object recognition system where the output of detection phase was as input for object recognition. In this paper, we have shown that the accuracy of object recognition is more than 95% as the object detection phase of our proposed algorithm detects a noise free ROI containing each object from any image. One of the major contribution of our model is achieving a comparatively higher rate of successful object detection and object classification than the state of art works.

References

[1] S. Ren, *et al.*, "Faster R-CNN: Towards real-time object detection with region proposal networks," *Advances in Neural Information Processing Systems*, 2015.

[2] J. Redmon, *et al.*, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[3] R. Girshick, *et al.*, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[4] A. Borji, *et al.*, "Salient object detection: A benchmark," *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 5706-5722, 2015.
[5] A. Angelova and S. Zhu, "Efficient object detection and segmentation for fine-grained recognition," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2013.
[6] Deformable part models. Ross Girshick UC Berkeley, CS231B Stanford University Guest Lecture. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.650.1184&rep=rep1&type=pdf
[7] M. E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *Proc. ICVGIP*, 2008.
[8] B. Leibe and B. Schiele, "Analyzing appearance and contour based methods for object categorization," in *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2003, vol. 2.
[9] D. Erhan, *et al.*, "Scalable object detection using deep neural networks," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2014.
[10] C. Szegedy, *et al.*, "Scalable, high-quality object detection," arXiv preprint arXiv:1412.1441, 2014.
[11] J. L. Pech-Pacheco, *et al.*, "Diatom autofocusing in brightfield microscopy: A comparative study," in *Proc. 15th International Conference on Pattern Recognition*, 2000, vol. 3.
[12] A. T. Young. Blurring and Unblurring. [Online]. Available: http://aty.sdsu.edu/bibliog/latex/scan/blur.html
[13] Sobel operator, sobel–feldman operator or filter. [Online]. Available: https://en.wikipedia.org/wiki/Sobel_operaton
[14] Image thresholding generated. [Online]. Available: https://docs.opencv.org/3.3.1/d7/d4d/tutorial_py_thresholding.html
[15] Converting from RGB to HSV. [Online]. Available: http://coecsl.ece.illinois.edu/ge423/spring05/group8/finalproject/hsvwriteup.pdf
[16] OpenCV, intel corporation, willow garage, Itseez. [Online]. Available: https://en.wikipedia.org/wiki/OpenCV
[17] Non-Photorealistic rendering using OpenCV (Python, C++). [Online]. Available: https://www.learnopencv.com/non-photorealistic-rendering-using-opencv-python-c/
[18] OpenCV threshold (Python, C++). [Online]. Available: https://www.learnopencv.com/opencv-threshold-python-cpp/
[19] M. Stricker and M. Orengo, "Similarity of color images," in Proc. *SPIE Conference on Storage and Retrieval for Image and Video Databases III*, 1995, vol. 2420, pp. 381-392.
[20] S. M. M. Islam and R. Debnath, "An RST invariant image retrieval approach using color moments and wavelet packet entropy," in *Proc. 5th International Conference on Informatics, Electronics and Vision*, 2016.
[21] S. Pertuz, D. Puig, and M. A. Garcia, "Analysis of focus measure operators for shape-from-focus," *Pattern Recognition*, vol. 46, no. 5, pp. 1415-1432, 2013.
[22] The PASCAL visual object classes. [Online]. Available: http://host.robots.ox.ac.uk/pascal/VOC/voc2007/
[23] I. Endres and D. Hoiem. (2010). Category independent object proposals. *Proc. ECCV*. [Online]. Available: https://en.wikipedia.org/wiki/OpenCV

Rafflesia Khan is a student of MS (Computer Science and Engineering) at Computer Science and Engineering discipline, University of Khulna, Khulna, Bangladesh. She has completed her Bachelor degree from Computer Science and Engineering discipline, Khulna University, Khulna, Bangladesh.in 2017. Her research areas of interest are Image classification, Object detection & recognition, Machine learning, and Image processing.

Tarannum Fariha Raisa has completed her Bachelor degree from Computer Science and Engineering discipline, Khulna University, Khulna, Bangladesh.in 2017. Her research areas of interest are Image classification, Object detection & recognition, Machine learning, and Image processing.

Professor Dr. Rameswar Debnath is a Professor of Computer Science and Engineering discipline at Khulna University, Khulna, Bangladesh. He has completed his Bachelor degree from Computer Science and Engineering discipline, University of Khulna, Khulna, Bangladesh.in 1997. He has received his PhD degree in Computer Science and Engineering from the University of Electro-Communications, Tokyo, Japan in 2005. His research areas of interest are Image classification, Object detection, Image processing, statistical machine learning (in particular supervised learning; support vector machines and kernel methods), and its applications to pattern recognition, image processing, bioinformatics and natural language analysis.