

DETECTION AND CONTOURING OF BAU-KUL USING IMAGE PROCESSING TECHNIQUES

M. M. Rahman^{1*} and M. M. H. Oliver¹

Abstract

Automated grading and sorting of fruits during harvesting period are needed for securing better market prices. In order to introduce such automation facilities in Bangladesh, edging and contouring information of the locally grown fruits is important. This study reports the first endeavor towards the use of image processing techniques for a popular jujube variety (BAU-Kul) in Bangladesh. Image processing techniques were used for segmentation, and contouring on the basis of color *Thresholding*, edge detection and contour detection in *Python-OpenCV* software. Six random samples of BAU-Kul fruit were used for the research. Perimeter lengths obtained from the image analysis of the six samples ranged from 17.9 cm to 20.20 cm with an average of 19.29 (± 1.02) cm. The measured lengths on the other hand, varied from 16.2 cm to 19.1 cm with an average of 17.75 (± 1.3) cm. Consequently, the average error in calculation was limited to only 7.98%. This indicates the fact that images captured through mobile devices can be used for detection and contouring of BAU-Kul samples with fairly high accuracy (92.02%). These information provides a foreground basis of automation for the grading and sorting systems of BAU-Kul fruits in Bangladesh.

Keywords: BAU-Kul, contour detection, edge detection, image segmentation.

Introduction

Jujube (baroi) is one of the popular fruits containing vitamin A and C. It is used in different food preparations such as Jam, Jelly, Chatney, Pickles and Juice. There are many jujube varieties that reportedly contains 85.9% water, 0.8% protein, 0.1% fat, 12.8% carbohydrate, 0.03% calcium, 0.03% phosphorus and 0.8% iron (Uddin and Hussain, 2012). In Bangladesh, popular varieties include Apple kul, BAU kul, BARI kul, Narkeli, and Sabzi. This study particularly deals with BAU-Kul developed by the Germplasm Center at Bangladesh Agricultural University. This

variety is most popular in the country because of its attractive size and texture. It is widely grown in Bangladesh ranging from sandy to saline and hilly and char land areas (Rahman and Islam, 2013; KGF, 2014). As with the other fresh produces, the market value of BAU-Kul depends on its sizes and color features. Although graders are available to separate products based on their sizes, no grading system based on the shape and colorimetric automation is available in Bangladesh. In particular, scientific information relating to separation and grading of BAU-Kul is not available in the literature.

¹Department of Agricultural Engineering, Bangabandhu Sheikh Mujibur Rahman Agricultural University, Gazipur 1706, Bangladesh. *Corresponding author: mostafizar-age@bsmrau.edu.bd

In order to develop a real time, nondestructive, and automated grading-sorting system for any fruit, the factorial combination of the size (detection, contouring and edging), and color (segmentation) is important. This technology requires the help of advance image processing techniques that include color space model, color *Thresholding*, edge detection and contour detection. Color is a property of an individual object which comes from the visible light reflecting off the object surface. In combination with color, Hue Saturation Value (HSV) space model are often used to locate the defects on fruits' surface in agricultural fields (Phakade *et al.*, 2014). There has been reports (Blasco *et al.*, 2009; Lin *et al.*, 2011) that color *Thresholding* can be used for the segmentation process of foreground images. Recent developments in automation has also experienced the use of *Canny Edge* detection method for the detection of edges in an image (Choudhary *et al.*, 2017; Rajani and Veena, 2019).

In such case, contour detection techniques are increasingly being used for analyzing noisy (Abubakar, 2013) and medical images (Senthilkumaran and Vaithegi, 2016). Similar approaches for computer vision technology for fruits (Feng and Qixin, 2004; Mahendran *et al.*, 2012; Nandi *et al.*, 2016; Sahu and Potdar, 2017) and vegetables (Deng *et al.*,

2017; Deulkar and Barve, 2018) have also been reported in the literature. More recently, a detailed contour based approach has been described by Septiarini *et al.* (2019). This emerging science of imaging has a potential application in the agricultural sector particularly, in the automatic grading (Banot and Mahajan, 2016; Nandi *et al.*, 2016; Deulkar and Barve, 2018) and sorting of agricultural products. In order for automation of this sector in Bangladesh, computer vision and related researches are required. Despite being a promising technology, very few scientific studies of this kind have been carried out in Bangladesh. In particular, many uniquely shaped and colored fruits of Bangladesh (for instance, BAU-Kul) have not been studied using advanced techniques. This study is going to shed some light on this area by employing color segmentation and contour detection approaches for BAU-Kul images. In order to achieve this, necessary algorithms will be generated using *Python-OpenCV* as recommended by Devi *et al.* (2017), Koirala *et al.* (2019) and Yonekura *et al.* (2019). The outcome of this research will provide a substantial basis in the development of an automated grading-sorting system for BAU-Kul products in Bangladesh.

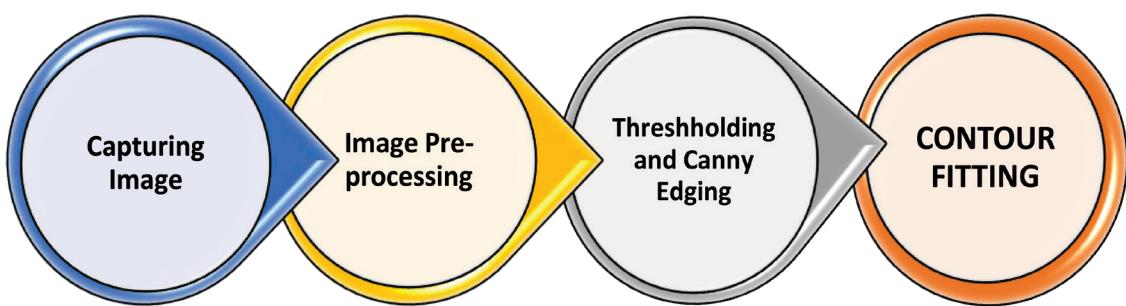


Fig. 1. Overall architecture behind BAU Kul detection and contouring.

Materials and Methods

The overall process of the BAU-Kul detection system has been illustrated in the following (Fig. 1) flowchart. Theoretically, the process of BAU-Kul detection system involves low-level processing and high level processing. In low-level processing, the digital color images were captured using a portable mobile devices. The captured RGB images were converted to HSV, and later to grayscale images in order to extract the *Thresholding* images.

In high level processing, the image was then processed through the *Canny Edge* function in order to obtain the parametric shapes of BAU-Kul samples. At the final step, the *Thresholding* and *Canny Edge* images were used to obtain the desired fits of contours around the BAU-Kul samples (Table 1).

The implementation of algorithm for contour fitting of BAU-Kul therefore, comprises of

the following consecutive steps i.e., (a) image acquisition (b) pre-processing (c) thresholding followed by canny edge detection, and (d) contouring. These steps and their mathematical models have been described as follows:

(a) Image acquisition

Samples of BAU-Kul were collected from the local market in Dhaka city in the month of February, 2019. Six randomly collected BAU-Kul samples were pictured using mobile devices. A description of the experimental set up has been summarized in Table 2.

The samples were laid out on a non-reflective surface, and naturally diffused sunlight (2-5 W/m²) was used for capturing these images. The device was set a fixed height so as to keep the focal length within (26-33 mm) for all the images so that the shadow effects could be minimized. The captured images were then saved as a .jpg/.jpeg format for further processing.

Table 1. BAU-Kul detection and contouring algorithm

Algorithm	Actions
Start	
Step-1:	Read BAU-Kul image into the <i>Python-OpenCV</i> Integrated Development Environment (IDE) from the particular folder.
Step-2:	Convert RGB image into the HSV color and Gray color.
Step-3:	Thresholding images by converting grayscale image into binary image and Canny edge detector uses for edge detection of BAU-Kul
Step-4:	Contour fitting and determine the contour perimeter
Stop	

Table 2. Experimental set up for image acquisition

Properties	Value	Properties	Value
Exposure value	0	Shutter speed (sec)	1/33
Color regime	RGB	Focal length (mm)	26-33 (equivalent to 35 mm focal length film)
White balance	AWB	ISO	400-500
Sample image size (pixel)	3456 × 4608	F-stop	f/2.2



BAU Kul Sample-1



BAU Kul Sample-2

Fig. 2. BAU-Kul samples for image processing.

(b) Image Pre-processing

The captured images were processed in two consecutive phases. In the first phase, *Python-OpenCV* computer languages were used to process the BAU-Kul color (*RGB*) images for detection and contouring. Normally, *RGB* (Red Green Blue) defines color in terms of a combination of primary colors,

```
# Image reading from files
image = cv2.imread ('E:/BAU-Kul sample-1.jpg')

# Getting green HSV color representation
hsv_img = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
cv2.imshow('HSV_Image_1', hsv_img)

# Converting the HSV image to Grayscale images
RGB_again = cv2.cvtColor(hsv_img, cv2.COLOR_HSV2RGB)
gray = cv2.cvtColor(RGB_again, cv2.COLOR_RGB2GRAY)
cv2.imshow('Gray_Image', gray)
```

where color information of the image is not separated from luminance. In contrast to *RGB*, *HSV* is used to separate image luminance from color information. That why, the *RGB* color images of the products were read into the *Python-OpenCV* Integrated Development Environment (IDE) and converted into the *HSV* color. The *HSV* model describes the colors similar to how human eyes tend to perceive color (Dash *et al.*, 2017) and is often preferred over the *RGB* model. Use of *HSV* model is particularly chosen in situations where color description plays an integral role. In this model, '*Hue*' represents the color, '*Saturation*' represents the amount to which that respective color is mixed with white, and '*Value*' represents the amount to which that respective color is mixed with black (*Gray level*).

(c) Thresholding and Canny edge detection

The BAU-Kul sample images were segmented using *Canny Edge* and *Thresholding*.

```
# Thresholding function
ret, threshold = cv2.threshold(gray, 90, 255, 0)
cv2.imshow('Threshold Image', threshold)
```

$$T = T[x, y, p(x, y), f(x, y)] \dots \quad (1)$$

where,

$f(x, y)$ is the gray level point of (x, y) , and

$p(x, y)$ indicates some local properties of this point.

The threshold image is defined as, $g(x, y) = 1$ if $f(x, y) > T$, and $g(x, y) = 0$ if $f(x, y) \leq T$. Usually, if the image intensity, $f(x, y)$ is less than a set threshold value of T , it is assumed to be a black pixel (Shapiro and George, 2002). Any higher value of $f(x, y)$ otherwise yields a white dot.

The action of *Thresholding* image segmented the image and produced binary images from a grayscale image. This is most effective in images with high levels of contrast (Shapiro George, 2002). The thresholding is used as an operation which involves tests against a function of threshold T of the form:

```
# Canny edge detection function
(ret, thresh) = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)
edge = cv2.Canny(thresh, 100, 200)
cv2.imshow('Canny Edge Image', edge)
```

In addition, *Canny Edge* detection is also an image segmentation technique, which extracts useful structural information (Muthukrishnan and Radha, 2011) from different vision objects, and reduce dramatically the amount of data to be processed. This method is quite complex and have five-step processes (Canny, 1986) that

(i) de-noises the image with a 5×5 Gaussian filter:

$$g(m, n) = G_\sigma(m, n) * f(m, n), \text{ where } G_\sigma = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{m^2 + n^2}{2\sigma^2}\right) \dots \quad (2)$$

(ii) calculates edge gradients and direction for each pixel:

$$G(m, n) = \sqrt{g_m^2(m, n) + g_n^2(m, n)}, \text{ and } \theta(m, n) = \tan^{-1} \left[\frac{g_n(m, n)}{g_m(m, n)} \right] \dots \quad (3)$$

(iii) applies non-maximum suppression (NMS) on edges obtained to thin out the edge ridges, and

(iv) sets a double threshold on all the detected edges to eliminate false positives.

It also analyzes all the edges and their connection to each other to keep the real edges and discard weaker ones (Minichino and Howse, 2015). In *OpenCV*, a single function is used to complete the whole process which is called *cv2.Canny()*. During the edge detection, the points where the intensity of colors changes significantly are found, and turned on, while turning the rest of the pixels off. The edge pixels are in an image, and there is no particular requirement that the pixels representing an edge are all contiguous.

```

# Contour detection
# function and Contour
# approximation
# function
(contours, _) = cv2.
findContours(edge,
copy(), cv2.RETR_
EXTERNAL, cv2.CHAIN_
APPROX_SIMPLE)
total = 0

# for contour in
contours:
    epsilon = 0.001 *cv2.
arcLength(contour,
True) approx = cv2.
approxPolyDP(contour,
epsilon, True)
cv2.
drawContours(image,
[approx], -1, (0, 0,
255), 3)
total += 1
print ("I found {0}
RET in that image",
format(total))
cv2.imshow('Contour_
Image', image)

# Calculation of
# contour perimeter
perimeter = cv2.
arcLength(cnt, True)
print("Perimeter of
contour:", perimeter)

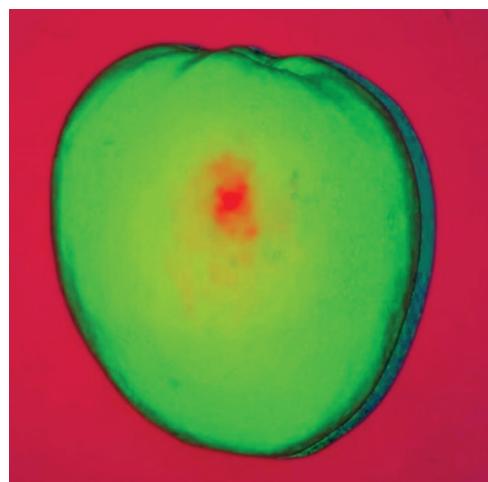
```

(d) Contour and contour perimeter

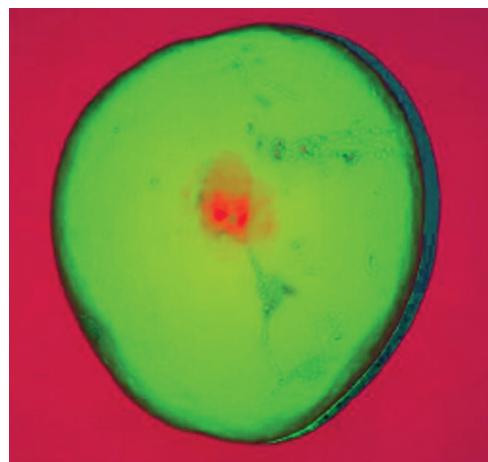
Finally, *cv2.find Contours()* and *cv2.draw Contours()* and functions were used to locate and visualize the contours in BAU-Kul images. Contour perimeter or curve length were also computed using the *cv2.arcLength()* function (Koirala *et al.*, 2019) which will evaluate the accuracy of manually determined perimeter with computed one.

Results and Discussion

HSV images produced from RGB originals of BAU-Kul sample image 1 and 2 have been shown in Fig. 3. This image is best for using the colors interactively. The HSV values are (143°, 8.2 %, 77%) for BAU-Kul sample 1 and (143°, 8.6%, 73%) for BAU-Kul sample 2 while the RGB color values of original images 1 and 2 were counted to be (179, 195, 185) and (169, 185, 175), respectively.



HSV Image for Bau-Kul sample-1



HSV Image for BAU-Kul sample-2

Fig. 3. HSV images for BAU Kul samples.

Usually, HSV is much easier for a user to obtain a desired color as compared to using RGB (Poorani *et al.*, 2013). *Thresholding* technique of BAU-Kul images isolates objects by converting grayscale images into binary images are shown in Fig. 4.



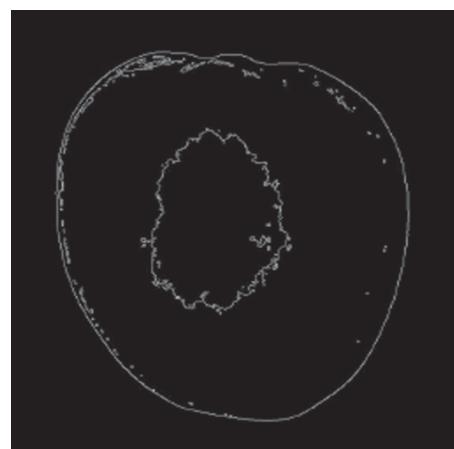
Threshold Image for Bau-Kul sample-1



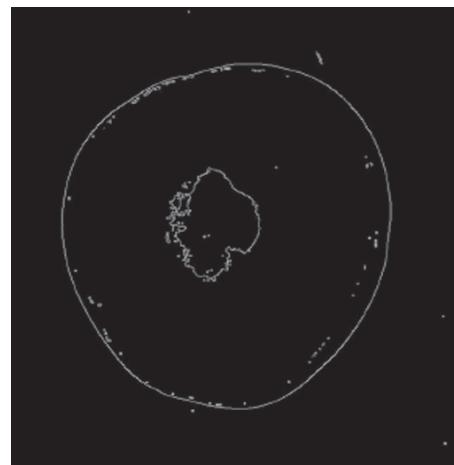
Threshold Image for Bau-Kul sample-2

Fig. 4. Thresholding images for BAU Kul samples.

This results showed that BAU-Kul image transformed into partially black and white, and the background of that images is transformed into completely white. In such kinds of thresholding techniques, the darker region (black and white) usually indicates the foreground (Efford, 2000) while the brighter (white) region is noted as the background. This however, would be useful for edge detection of the products as shown in Fig. 5.



Canny Edge for Bau-Kul sample-1



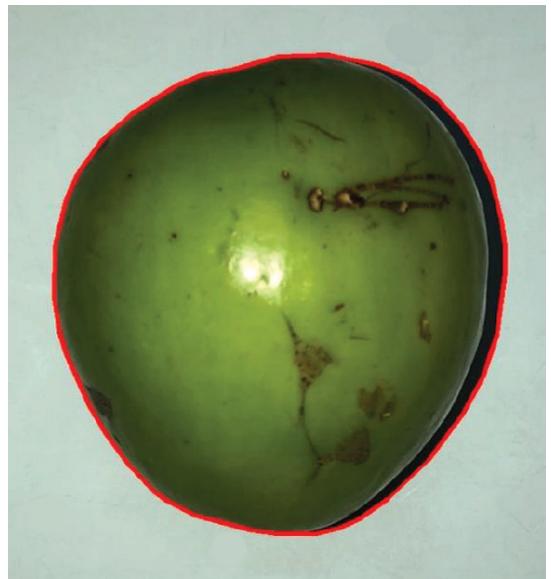
Canny Edge for Bau-Kul sample-2

Fig. 5. Canny Edge images for BAU Kul samples.

The results of *Canny Edging* of the input BAU-Kul samples (Fig. 5) are in fact, binary images in which the white pixels closely approximate the true edges of the original BAU-Kul product. However, some white dots were also identified in the middle of the image which apparently occurred due to the natural lights reflecting from the surface of the BAU-Kul sample when images were captured using mobile devices. The red color contours are obtained on the original BAU-Kul images (Fig. 6), which provides the line segments corresponding to the shapes of the objects in the images. If the lightings during images capturing are properly made, the shadow of the fruits on the surface could be minimized, and the contour of BAU-Kul will be more accurate. This manuscript however, present a simple technique in which images captured from normal mobile devices can be used for such kinds of operation. In case of commercial operations however, proper lighting arrangements can significantly improve the accuracy of edge detection.



Contour for Bau-Kul sample-1



Contour for Bau-Kul sample-2

Fig. 6. Contour images for BAU Kul samples.

Perimeter Quantification

The algorithm was also employed in computing the perimeter of all the six BAU-Kul samples as shown in Table 3. The obtained pixelated values were transformed into linear lengths in centimeter. In order for accurate calculation, a standard square of known dimensions (4 cm \times 4 cm) were used for calibration purpose in each case. In addition, the actual lengths of the samples around their exact photographed edges were measured using a graduated scale with ± 0.1 cm accuracy.

The difference between the measured and the calculated values were termed and expressed as % errors. As can be seen from Table 3, the algorithm employed under this manuscript was able to predict the perimeters of the BAU-Kul samples with considerable accuracy with errors ranging from 6.33 to 10.13%. The calculated lengths of the six samples ranged from 17.9 cm to 20.20 cm with an average

Table 3. Evaluation of contour perimeter for selected BAU-Kul samples

Samples	Calculated Perimeter (cm)	Measured Perimeter (cm)	% Error
S1	17.90	16.20	9.49
S2	20.20	18.90	6.46
S3	19.36	17.40	10.13
S4	20.39	19.10	6.33
S5	19.67	18.00	8.49
S6	18.22	16.90	7.22

($\pm SD$) of $19.29(\pm 1.02)$ cm. The measured lengths on the other hand, varied from 16.2 cm to 19.1 cm with an average ($\pm SD$) of $17.75(\pm 1.3)$ cm. Consequently, the average error was limited to only 7.98% (± 1.02). This indicates the fact that the algorithms employed

in this manuscript can be used for determining the BAU-Kul samples (Table 3) with considerable accuracy (92.02%) using images from mobile devices. If image acquisition can be done in a properly designed chamber, the accuracy can be substantially increased.

Conclusions

Image processing techniques are used to measure the qualities of fruits based on color space model, *Thresholding*, edge and contour detection methods. In this research, *Thresholding* segmentation, edge and contour detection were obtained from the color images of BAU-Kul by using *Python-OpenCV* platform. The edge and contour of BAU-Kul images were found to be good enough to be used for operational purposes. The contours will however, be more accurate if the suitable lighting conditions are employed in order to minimize the shadow of images on the background surfaces. Nonetheless, the algorithms employed in this manuscript can be used for determining the BAU-Kul samples with considerable accuracy (92.02%). The information generated from this research will be helpful for commercial grading and sorting facilities thriving for automation. This would ultimately help the growers get better price according to the shape and sizes of BAU-Kul.

References

- Abubakar, F. M. 2013. Study of Image segmentation using thresholding technique on a noisy image. *Int. J. Sci. Res.* 2(1): 49-51.
- Banot, S. and P. M. Mahajan. 2016. A fruit detecting and grading system based on image processing-review. *Int. J. Innov. Res. Electric. Electron. Instr. Contr. Eng.* 4(1): 47-52.
- Blasco, J., N. Aleixos, S. Cubero, J. Gómez-Sanchís and E. Moltó. 2009. Automatic sorting of satsuma (citrusunshiu) segments using computer vision and morphological features. *Comput. Electron. Agric.* 66(1): 1-8.
- Canny, J. 1986. A computational approach to edge detection. *IEEE Trans. Pattern Analys. Mach. Intel.* 8(6): 679-698.
- Choudhary, P., R. Khandekar, A. Borkar and P. Chotaliya. 2017. Image processing algorithm for fruit identification. *Int. Res. J. Eng. Technol.* 4(3): 2741-2743.
- Dash, S. S., P. C. B. Naidu, R. Bayindir and S. Das. 2017. Artificial intelligence and evolutionary computations in engineering systems. *Proc. ICAIECES*. Springer Nature Singapore, 468 P.
- Deng, L., H. Du and Z. Han. 2017. A carrot sorting system using machine vision technique. *Appl. Eng. Agric.* 33(2): 149-156.
- Deulkar, S. S. and S. S. Barve. 2018. An automated tomato quality grading using clustering based support vector machine. *3rd Int. Conf. Communic. Electron. Syst.* Pp. 1128-1133.
- Devi, T. G., P. Neelamegam and S. Sudha. 2017. Image processing system for automatic segmentation and yield prediction of fruits using open CV. *Int. Conf. Curr. Trend. in Comp. Electric. Electron. Communic.* Pp. 758-762.
- Efford, N. 2000. Segmentation. Pp. 250-270. *Digital Image Processing: A Practical Introduction Using Java*. Pearson education, England.
- Feng, G. R. and C. Qixin. 2004. Study on color image processing based intelligent fruit sorting system, *Fifth World Congress on Intel. Contr. Auto.* 6 (6): 4802-4805.
- Kanimozhi, B. and R. Malliga. 2017. Classification of ripe or unripe orange fruits using the color coding technique. *Asian J. Appl. Sci. Technol.* 1(3): 43-47.
- Koirala, A., K. B. Walsh, Z. Wang, and C. McCarthy. 2019. Deep learning for real-time fruit detection and orchard fruit load

- estimation: Benchmarking of mango YOLO. *Precis. Agric.* 20(6): 1107-1135.
- KGF. 2014. Annual Report (Jan-Dec, 2014), Krishi Gobeshona Foundation, 26 P.
- Lin, C., C. H. Su, H. S. Huang and K. C. Fan. 2011. Colour image segmentation using relative values of RGB in various illumination circumstances. *Int. J. Com.* 5(2): 252-261.
- Mahendran, R., G. C. Jayashree and K. Alagusundaram. 2012. Application of computer vision technique on sorting and grading of fruits and vegetables. *J. Food Process Technol.* S1-001.
- Minichino, J. and J. Howse. 2015. *Learning OpenCV 3 computer vision with python*. 2nd Edition. Packt publishing ltd. livery place, Birmingham, UK. 55 P.
- Muthukrishnan, R. and M. Radha. 2011. Edge detection techniques for image segmentation. *Int. J. Comp. Sci. Info. Technol.* 3(6): 259-267.
- Nandi, C. S., B. Tudu and C. Koley. 2016. A machine vision technique for grading of harvested mangoes based on maturity and quality. *IEEE Sensors J.* 16(16): 6387-6396.
- Phakade, S. V., D. Flora, H. Malashree and J. Rashmi. 2014. Automatic fruit defect detection using HSV and RGB color space model. *Int. J. Innov. Res. Comp. Sci. Technol.* 2(3): 67-73.
- Poorani, M., T. Prathiba and G. Ravindran. 2013. Integrated feature extraction for image retrieval. *Int. J. Comp. Sci. Mob. Comp.* 2(2): 28-35.
- Rahman, M. M., and A. Islam. 2013. Adaptation technologies in practice and future potentials in Bangladesh. *Climate Change Adaptation Actions in Bangladesh*. Springer, Tokyo. Pp. 305-330.
- Rajani, S. and M. N. Veena. 2019. Medicinal plants segmentation using thresholding and edge based techniques. *Int. J. Innov. Technol. Explor. Eng.* 8(6S4): 71-76.
- Sahu, D. and R. M. Potdar. 2017. Defect identification and maturity detection of mango fruits using image analysis. *Americ. J. Art. Intel.* 1(1): 5-14.
- Senthilkumaran, N. and S. Vaithogi. 2016. Image segmentation by using thresholding techniques for medical images. *Comp. Sci. Eng.: An Int. J.* 6(1): 1-13.
- Septiarini, A., H. Hamdani, H. R. Hatta and K. Anwar. 2019. Automatic image segmentation of oil palm fruits by applying the contour-based approach. *Scientia Hort.* 261: 1-7.
- Shapiro, L. G. and C. S. George. 2002. *Computer Vision*. New jersey, Prentice hall. ISBN 0-13-030796-3.
- Uddin, M. B. and I. Hussain. 2012. Development of diversified technology for jujube (*Ziziphus jujuba* L) processing and preservation. *World J. Dairy Food Sci.* 7(1): 74-78.
- Yonekura, T., A. Iwamoto, H. Fujita, and M. Sugiyama. 2019. Mathematical model studies of the comprehensive generation of major and minor phyllotactic patterns in plants with a predominant focus on orixate phyllotaxis. *PLoS Comp. Biol.* 15(6), e1007044.

