Foreground algorithms for detection and extraction of an object in multimedia

Rekha V.[1], Natarajan K[2], Innila Rose J.[3]
[1,2]Department of Computer Science and Engineering, Faculty of Engineering, Christ (Deemed to be University), India
[3]Center for digital innovation (CDI), Faculty of Engineering, Christ (Deemed to be University), India.

Article Info

Article history:

Received Apr 1, 2019
Revised Oct 23, 2019
Accepted Nov 4, 2019

Keywords:

Background subtraction
Contours
Foreground subtraction
Grab cut
Haar-cascade
HOG
Image segmentation
Static background
Vab cut
Watershed

ABSTRACT

Background Subtraction of a foreground object in multimedia is one of the major preprocessing steps involved in many vision-based applications. The main logic for detecting moving objects from the video is difference of the current frame and a reference frame which is called “background image” and this method is known as frame differencing method. Background Subtraction is widely used for real-time motion gesture recognition to be used in gesture enabled items like vehicles or automated gadgets. It is also used in content-based video coding, traffic monitoring, object tracking, digital forensics and human-computer interaction. Now-a-days due to advent in technology it is noticed that most of the conferences, meetings and interviews are done on video calls. It’s quite obvious that a conference room like atmosphere is not always readily available at any point of time. To eradicate this issue, an efficient algorithm for foreground extraction in a multimedia on video calls is very much needed. This paper is not to just build Background Subtraction application for Mobile Platform but to optimize the existing OpenCV algorithm to work on limited resources on mobile platform without reducing the performance. In this paper, comparison of various foreground detection, extraction and feature detection algorithms are done on mobile platform using OpenCV. The set of experiments were conducted to appraise the efficiency of each algorithm over the other. The overall performances of these algorithms were compared on the basis of execution time, resolution and resources required.

Corresponding Author:

Innila Rose J.,
Center for digital innovation (CDI),
Faculty of Engineering,
Christ (Deemed to be University), India.
Email: Rose.innila.10@gmail.com

1. INTRODUCTION

Background subtraction is the means of detaching the foreground objects from the background in a progression of video frames. In the most recent two decades there have been a numerous, improvements in methods of doing background subtraction. This algorithm has broad usage in various crucial applications like visual monitoring, sports video judgement, action seizure, and so on [1-6]. In this paper a comparative study of all the foreground detection and extraction algorithms is presented. For detection Haar-cascade, contours, watershed and HOG method are compared. For extraction vabcut and grab-cut algorithms are compared. The main motive of this comparison is to ensure which algorithm is a better one and will provide a comparatively less time for execution than others since the main idea of this project is to execute the extraction process on a mobile platform and for that the execution time in desktop should be minimal. In this paper, foreground segmentation in a video is done using OpenCV. It was found that for foreground segmentation in OpenCV there is an inbuilt method of background subtraction. All the basic inbuilt methods for foreground segmentation [7] were implemented to enable a basic understanding of how the code works and to check whether making any minor changes in it affects the output variance. The aim was to find an efficient algorithm for foreground extraction in a multimedia.

The main objective of this work is not to just build Background Subtraction application for Mobile Platform but to optimize the existing OpenCV algorithm to work on limited resources on mobile platform without reducing the performance. The main objectives of the paper were as follows:

– To detect the foreground object and get the outline of the foreground object.
– To extract the foreground object effectively and varyresolutions of the video accordingly.
– Enable background replacement and Reduce the execution time.
– Blending the new background with the extracted foreground object.

In paper [8] CNN method is used for obtaining the foreground mask. They used the dataset of CDnet 2014 and are using Subsense algorithm to prevent over-fitting. In paper [9] selective background Subtraction is used. Background modelling is used to deduct two consecutive frames to get the static pixels. The proposed technique helps in withholding unwanted background from the forefront and the backdrop scene instead of subtracting the whole background. Paper [10] focuses on reducing the holes which is obtained while getting a foreground mask. This paper explains and implements why three frame differencing are used instead of two frame differencing**.** Paper [1] focuses on MOG method for detection of object. They focused on five criteria of MOG particularly segmented background weight threshold, standard deviation scaling factor, user defined learning rate, total number of Gaussian components and maximum number of components in the background model. Paper [11-16] is on video segmentation for extracting the foreground object in a video. In this paper they are using SURF algorithm for feature detection of the foreground object and eventually segmenting them effectively. Paper [17] talks about Grab Cut Algorithm which is one of the familiar approaches for foreground extraction in the domain of Image processing. In paper [1] FCFN mechanism is used for background subtraction. FCFN means fuzzy nearness degree which uses fuzzy c-means clustering algorithm for identifying whether the pixel selected is background or foreground.

2. RESEARCH METHOD

2.1. Frame work model of background subtraction of an object in multimedia

Background Subtraction of an object in multimedia (BSOM) is one of the primary steps in various view-based operations. As the name suggests background subtraction is the mechanism of deducting forefront substances from the environment in a series of video blocks. The main philosophy for identifying dynamic substances from the video is to subtract between the present frame and a reference frame which is called "background image" and this scheme is known as frame differencing method. Background Subtraction is extensively used in traffic surveying (detecting vehicles), human movement identification, human-machine co-operation, object movement tracing, digital forensics etc. In below Figure 1 Firstly, the video is divided into frames. After that subtraction of the estimated background i.e the previous frame from the current frame is done. Later on, a threshold is applied to get a clear foreground mask. Based on the threshold value we will get a clear foreground mask. It can be done either manually by giving a threshold value or by using various threshold algorithms.

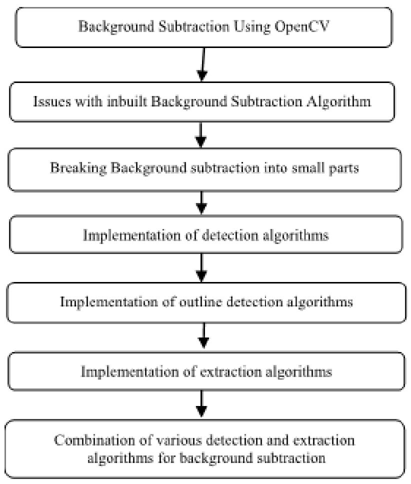
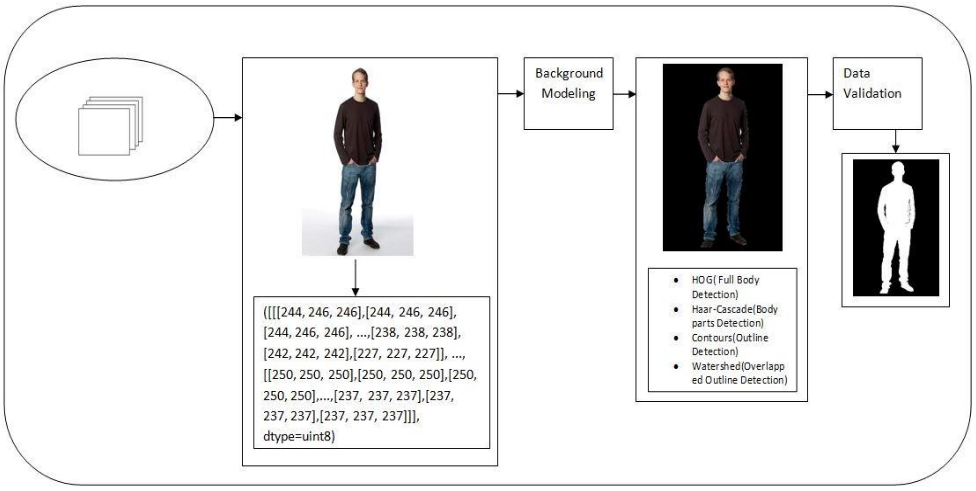

Background Subtraction Using OpenCV
Issues with inbuilt Background Subtraction Algorithm
Breaking Background subtraction into small parts
Implementation of detection algorithms
Implementation of outline detection algorithms
Implementation of extraction algorithms
Combination of various detection and extraction algorithms for background subtraction

Figure 1. Methodology

The four primary steps in Background Subtraction are Pre-processing, background modeling, foreground detection and data validation. In the first step, a given video is broken down into frames. Then the input frame is converted into numpy array for further processing. Then from this numpy array the background model is formed. If it's a video then two consecutive frames are taken and deducted to get the static pixels. These static pixels constitute the background and thus background model is formed. Further all the pixels in a frame are compared with the background model thus formed which helps in identifying the significant forefront pixels effectively. Lastly the unwanted pixels which neither constitutes the forefront nor the background are removed to get a proper foreground mask as shown in the last step of Figure 2.


Background Modeling
Data Validation
HOG(Full Body Detection)
Haar-Cascade(Body parts Detection)
Contours(Outline Detection)
Watershed(Overlapped Outline Detection)

```
([[[244, 246, 246],[244, 246, 246],
[244, 246, 246], ...,[238, 238, 238],
[242, 242, 242],[227, 227, 227]], ...,
[[250, 250, 250],[250, 250, 250],[250,
250, 250],...,[237, 237, 237],[237,
237, 237],[237, 237, 237]]],
dtype=uint8)
```

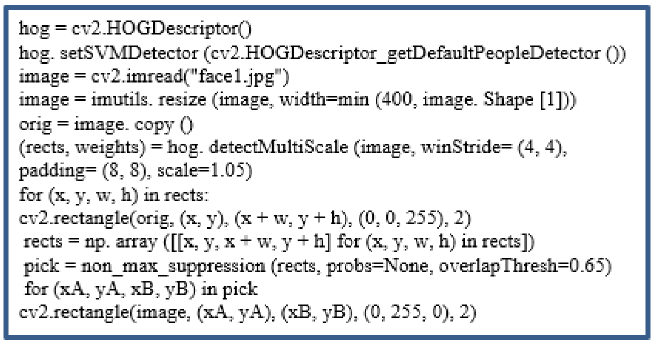
Figure 2. Block Diagram of Background Subtraction Process.

2.2. Techniques used for BSOM

In this paper, various techniques are used for BSOM for foreground object detection and extraction [18-21]. Following section of the paper discusses techniques and its implementation:

2.2.1. Histogram of gradients (HOG)

In Figure 3 the HOG algorithm is shown which uses SVMDetector for detecting the foreground object in an image or a video even when they are of different scales. After the human foreground is detected, a rectangle is formed around it and displayed as output.

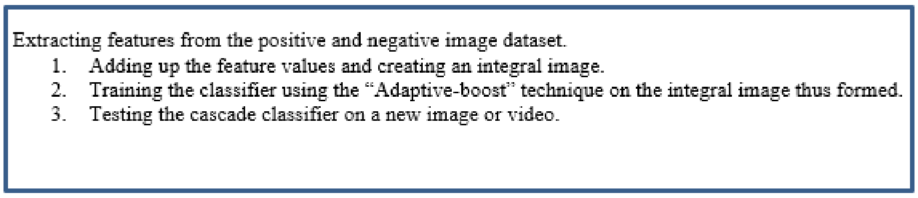


```
hog = cv2.HOGDescriptor()
hog. setSVMDetector (cv2.HOGDescriptor_getDefaultPeopleDetector ())
image = cv2.imread("face1.jpg")
image = imutils. resize (image, width=min (400, image. Shape [1]))
orig = image. copy ()
(rects, weights) = hog. detectMultiScale (image, winStride= (4, 4),
padding= (8, 8), scale=1.05)
for (x, y, w, h) in rects:
cv2.rectangle(orig, (x, y), (x + w, y + h), (0, 0, 255), 2)
 rects = np. array ([[x, y, x + w, y + h] for (x, y, w, h) in rects])
 pick = non_max_suppression (rects, probs=None, overlapThresh=0.65)
 for (xA, yA, xB, yB) in pick
cv2.rectangle(image, (xA, yA), (xB, yB), (0, 255, 0), 2)
```

Figure 3. HOG algorithm.

2.2.2. Haar-cascade algorithm

In Figure 4 Haar cascade is algorithm is shown step-wise. Suppose face detection is taken into consideration, then the input dataset will have images of faces in all its different views. Then the haar features are identified in those images and an integral image is formed as shown in Figure 5. Then these integral images are taken and given to various classifiers using the adaptive boosting technique. Finally, the machine is trained to identify faces in a certain image or video.



Extracting features from the positive and negative image dataset.

1. Adding up the feature values and creating an integral image.
2. Training the classifier using the "Adaptive-boost" technique on the integral image thus formed.
3. Testing the cascade classifier on a new image or video.

Figure 4. Haar-cascade algorithm [22]

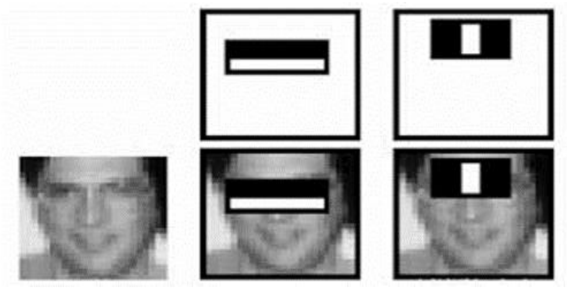
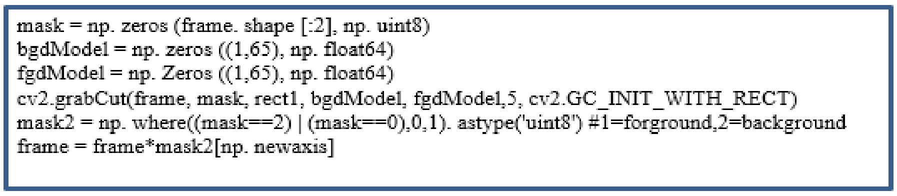


Figure 5. Integral Image [22]

2.2.3. Grab-cut algorithm

In Figure 6 Grab cut algorithm is shown where firstly a mask in applied to the inputted image for getting the foreground and background objects. Here if the mask value is 2 then it is considered as background and if the mask value is 1 then it is considered as foreground. Finally, this mask value is multiplied with the frame to get a foreground mask.

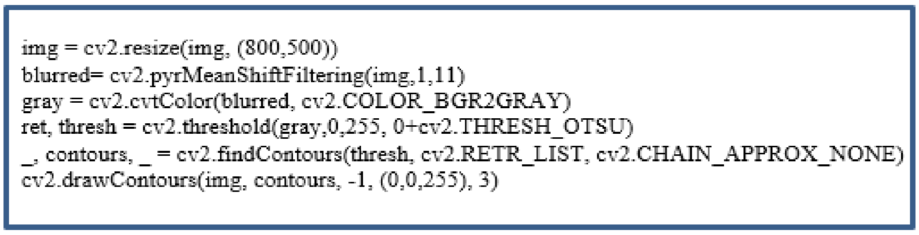


```
mask = np. zeros (frame. shape [:2], np. uint8)
bgdModel = np. zeros ((1,65), np. float64)
fgdModel = np. Zeros ((1,65), np. float64)
cv2.grabCut(frame, mask, rect1, bgdModel, fgdModel,5, cv2.GC_INIT_WITH_RECT)
mask2 = np. where((mask==2) | (mask==0),0,1). astype('uint8') #1=forground,2=background
frame = frame*mask2[np. newaxis]
```

Figure 6. Grab-cut algorithm [23]

2.2.4. Contour algorithm

In Figure 7 the algorithm of contours is given where resizing of image is done and then a pyr mean shift filter is applied to it. Then the image is converted to grayscale and some threshold value is given. An inbuilt function of findContours is used to identify the contours in the image and by using the drawContours function the identified contour is drawn and displayed in the output.

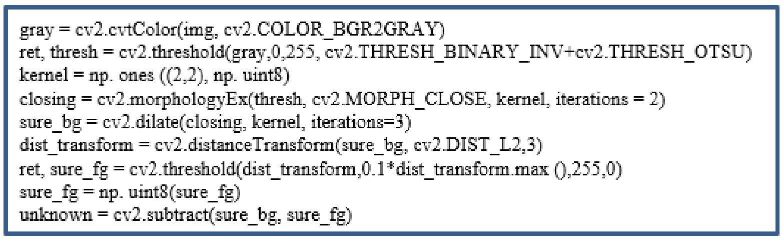


```
img = cv2.resize(img, (800,500))
blurred= cv2.pyrMeanShiftFiltering(img,1,11)
gray = cv2.cvtColor(blurred, cv2.COLOR_BGR2GRAY)
ret, thresh = cv2.threshold(gray,0,255, 0+cv2.THRESH_OTSU)
_, contours, _ = cv2.findContours(thresh, cv2.RETR_LIST, cv2.CHAIN_APPROX_NONE)
cv2.drawContours(img, contours, -1, (0,0,255), 3)
```

Figure 7. Contour algorithm [24]

2.2.5. Watershed algorithm

In Figure 8, watershed algorithm is given. The inputted image is converted into grayscale and then a suitable threshold value is applied to it. A suitable kernel size is taken and the morphological transformations are applied to it. The sure background and foreground are obtained and then they are subtracted to get the unknown region which is the boundary of the forefront object in the inputted image. From the above the Haar-cascade is used to detect the foreground object and the Grab-cut is used to extract the foreground thus detected.



```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
ret, thresh = cv2.threshold(gray,0,255, cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
kernel = np. ones ((2,2), np. uint8)
closing = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, kernel, iterations = 2)
sure_bg = cv2.dilate(closing, kernel, iterations=3)
dist_transform = cv2.distanceTransform(sure_bg, cv2.DIST_L2,3)
ret, sure_fg = cv2.threshold(dist_transform,0.1*dist_transform.max (),255,0)
sure_fg = np. uint8(sure_fg)
unknown = cv2.subtract(sure_bg, sure_fg)
```

Figure 8. Watershed algorithm

2.3. Analysis of various algorithms for BSOM

In this section of the paper gives detailed description of comparison of various algorithms. These algorithms are implemented for the BSOM.

– Comparison of Histogram of Gradients (HOG) and Haar-cascade for detection of foreground: HOG method is used to detect the full body of a human unlike Haar-cascade algorithm. Haar-cascade algorithm has various xml files for different parts of the body like half body, face, full body, smile etc. [22].
– Comparison of Grab-cut and Vab-cut: Grab Cut and Vab cut both are used in the field of image segmentation. Both the algorithm helps in extracting the foreground object. The drawback of the grab-cut algorithm is that it is not optimized. Vab Cut is an extension of Grab Cut [23].
– Comparison of Contours and Watershed: Contours and watershed are used for the detection of the outline of the foreground object from an image or a video. Contours uses the function "find Contours" to identify the contours in the given image or video. In watershed algorithm, morphological transforms are done on the image first and then sure foreground and background is identified [24, 25].

3. RESULTS AND DISCUSSIONS

In this work OpenCV tool was used which is an open source library that approves use of various computer languages and is applicable on several platforms. From the above Figures 9-11 it can be concluded that Haar-cascade is better detection algorithm as the input image or video doesn't always have to be full body. From Figures 12, 13 it can be concluded that Grab-cut isn't efficient in extracting the foreground object. Figure 14 clearly shows how manual masking the input image helps in the effective extraction using Grab-cut [22]. Figures 15 and 16 show how contours are drawn in an image and video. Figures 17 and 18 shows how watershed is different from contours.

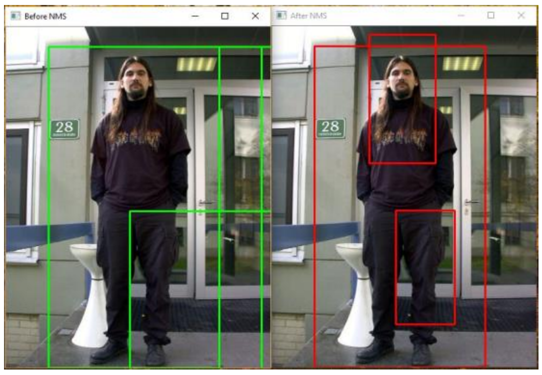


Figure 9. Non-Maximu suppression in HOG

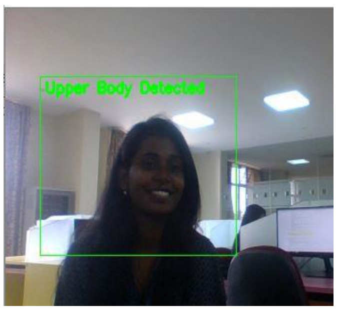


Figure 10. Haar-cascade of 2160p

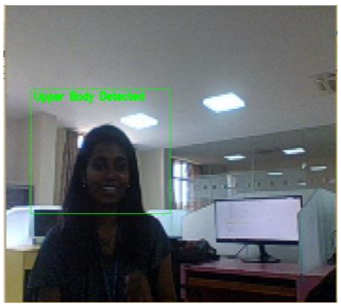


Figure 11. Haar-cascade of 144p

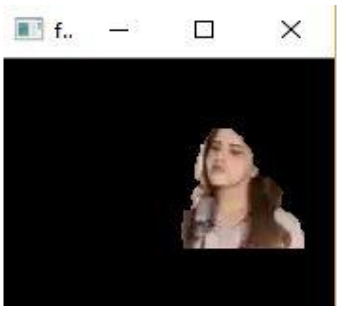


Figure 12. Grab-cut in a video in 480p

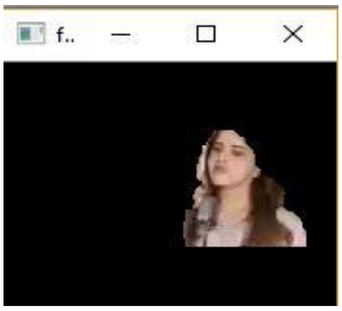
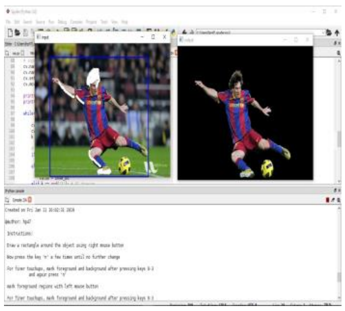


Figure 13. Garb-cut in a video in 720p



```
Created on Fri Jan 11 16:02:31 2019

@author: hp47

Instructions:

Draw a rectangle around the object using right mouse button

Now press the key 'n' a few times until no further change

For finer touchups, mark foreground and background after pressing keys 0-3
        and again press 'n'

mark foreground regions with left mouse button

For finer touchups, mark foreground and background after pressing keys 0-3
```

Figure 14. Grab-cut using manual masking on an image [23]

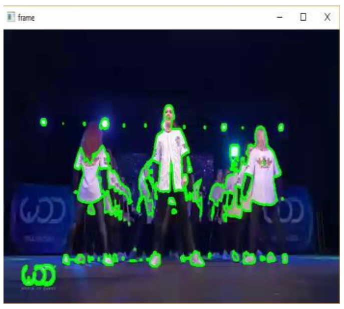


Figure 15. Contour in a video

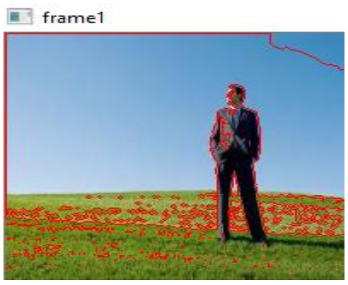


Figure 16. Contour in an image

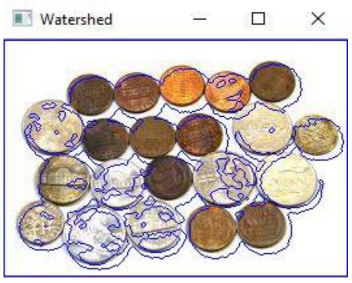


Figure 17. Watershed in an image [24]

In Figures 18 and 19 a difference in the output is seen when a certain video of a span of 30 seconds is downloaded in different resolutions, to check the effectiveness of the output. Tabular values of haar-cascade with grab-cut for a video downloaded at different resolutions are shown in Table 1 and graphical representation are shown in Figure 20. A video of 30 seconds with a certain resolution is taken, and a table of comparison of its execution time of each of the algorithm (used in the code) is shown in Table 2 with different resolution values (changed by the code) and graphical representation of Table 2 is shown in Figure 21. A video of 30 seconds with a certain resolution is taken, and a table of comparison of its execution time of each of the algorithm (used in the code) is shown with different resolution values (changed by the code) in Table 3 and Figure 22 shows the graphical represenatation of Table 3. After the survey of all these results it was found that Haar-cascade with Grab-cut gave a better result than any other combination. The results can be clearly seen in Figure 23 where a resolution of 144p is taken into consideration.

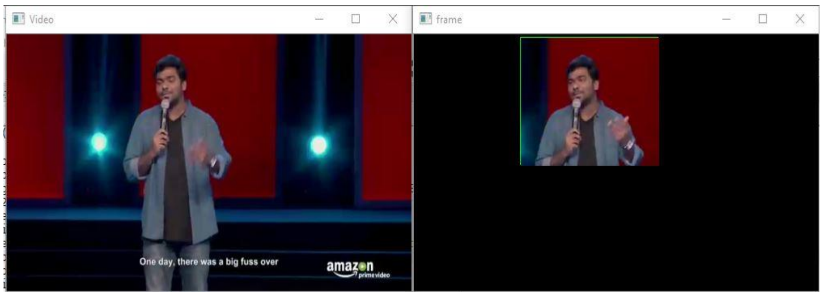


Figure 18. Haar-cascade with Grab-cut on a video downloaded with a 720p resolution

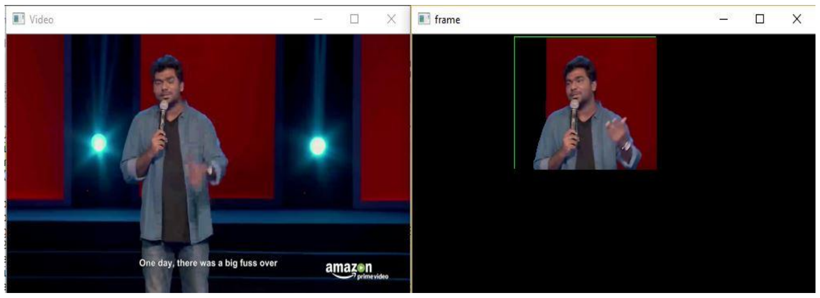


Figure 19. Haar-cascade with Grab-cut on a video downloaded with a 1080p resolution

Table 1. Tabular values of haar-cascade with grab-cut for a video downloaded at different resolutions

Resolution (In Pixels)	Haar-Cascade Algorithm with Grab-Cut Algorithm (In Seconds)
1080p	79.05
720p	60.91
360p	31.34
144p	378.13

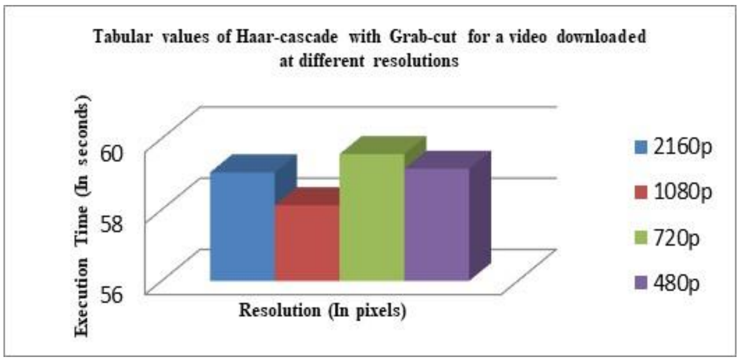


Figure 20. Graphical representation of the Table 1 in Figure 19

Table 2. Tabular values of haar-cascade with grab-cut for a video of certain resolution

Resolution (In Pixels)	Haar-Cascade Algorithm (In Seconds)	Grab-Cut Algorithm (In Seconds)	Haar-Cascade Algorithm with Grab-Cut Algorithm (In Seconds)
2160p	59.05	71.40	127.47
1080p	58.13	70.16	125.19
720p	59.57	67.54	142.20
480p	59.16	67.75	144.76
360p	58.92	66.54	124.87
240p	58.02	65.42	129.29
144p	59.05	66.58	129.51

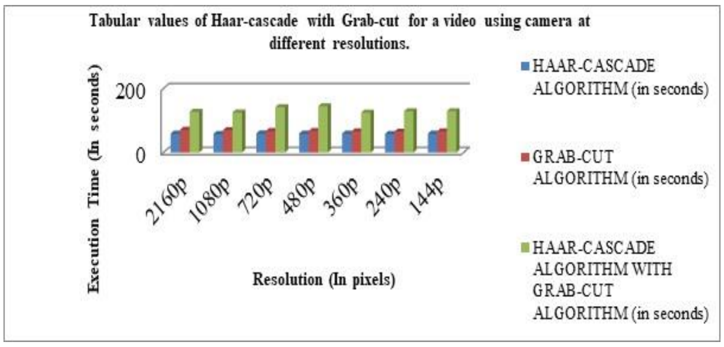


Figure 21. Graphical representation of the Table 2

Table 3. Tabular values of haar-cascade with grab-cut for a video of certain resolution

Resolution (In Pixels)	Haar-Cascade Algorithm (In Seconds)	Grab-Cut Algorithm (In Seconds)	Haar-Cascade Algorithm with Grab-Cut Algorithm (In Seconds)
2160p	59.05	71.40	127.47
1080p	58.13	70.16	125.19
720p	59.57	67.54	142.20
480p	59.16	67.75	144.76
360p	58.92	66.54	124.87
240p	58.02	65.42	129.29
144p	59.05	66.58	129.51

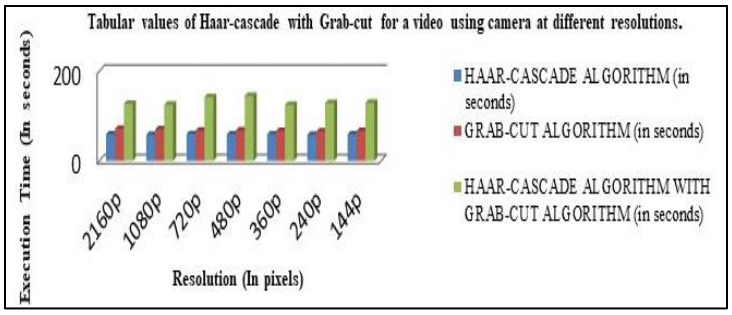
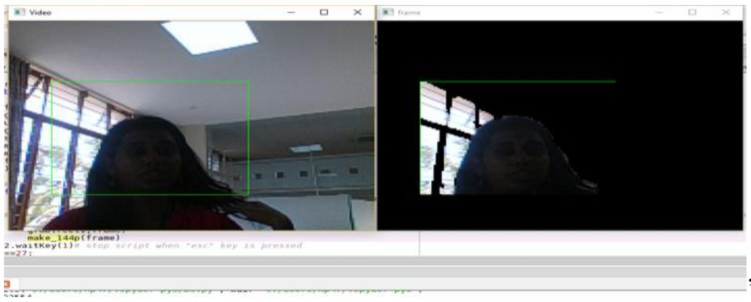


Figure 22. Graphical representation of the Table 3



```
    make_144p(frame)
2.waitKey(1)# stop script when "esc" key is pressed
==27:
```

,

Figure 23. Final output

4. CONCLUSION

In this paper, various foreground detection and extraction algorithms are compared. The main objective of our project is to identify various inbuilt methods for getting a best foreground mask. Out of these algorithms we found that the combination of Grab cut and Haar cascade is best for extracting the foreground object. As clearly seen in Figure 23 there is a live video taken from the camera itself at a resolution of 144p and grab-cut is applied to it. The only drawback of this method is that the light coming from the window (as shown is the figure) cannot be cut out using grab-cut. For further work we can focus on optimizing the grab cut algorithm for extraction of foreground object even when a light source is near it.

REFERENCES

[1] Shahrizat Shaik Mohamed, N. M. "Background Modelling and Background Subtraction Performance for Object Detection," *2010 6th International Colloquium on Signal Processing & its Applications,* Mallaca City, pp. 1-6. 2010.

[2] Sen-Ching S. Cheung and Chandrika Kamath, "Robust Techniques for Background Subtraction in Urban Traffic Video," *EURASIP Journal on Applied Signal Processing,* vol. 2005, pp. 2330-2340, Jan. 2005.

[3] F. El Baf, T. Bouwmans, and B. Vachon. "Comparison of background subtraction methods for a multimedia application," *International Conference on systems, Signals and Image Processing, IWSSIP 2007,* pages 385–388, June 2007.

[4] Donovan H. Parks and Sidney S. Fels, "Evaluation of Background Subtraction Algorithm with Post-Processing," in *IEEE Fifth International Conference on Advanced Video & Signal Based Surveillance,* p.192-199, 2008.

[5] Massimo Piccardi," Background Subtraction Techniques: A Review," *IEEE International Journal on Systems, Man and Cybernetics,* Vol. 2, (5), pp. 05-25, 2004.

[6] Gourav Takhar, Chandra Prakash, Namita Mittal, Rajesh Kumar, "Comparative Analysis of Background Subtraction Techniques and Applications," *IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2016),* pp. 1-8, 2016.

[7] S. Hore, et al., "An integrated interactive technique for image segmentation using stack based seeded region growing and thresholding," *International Journal of Electrical and Computer Engineering,* vol/issue: 6(6), pp. 2773–2780, 2016.

[8] Chuming Lin, B. Y, "Foreground Detection in Surveillance Video with Fully Convolutional Semantic Network," *2018 25th IEEE International Conference on Image Processing (ICIP),* Athens, pp. 4118-4122, 2018.

[9] Neha S. Sakpal, M. S, "Adaptive Background Subtraction in Images", *IEEE Transactions on Multimedia,* 18(10), pp. 2093 – 2103, 2018.

[10] Rahul Dutt Sharma, S. L. "Optimized Dynamic Background Subtraction Technique for Moving Object Detection and Tracking," *2017 2nd International Conference on Telecommunication and Networks (TEL-NET),* Noida, pp. 1-3. 2017.

[11] Yanxin Sun, G.Y., "The Foreground Segmentation Based on Surf Algorithm and Background Subtraction," *2015 Seventh International Conference on Advanced Communication and Networking (ACN),* Kota Kinabalu, pp. 24-27. 2015.

[12] Carsten Rother and Vladimir Kolmogorov and Andrew Blake, ""GrabCut": Interactive Foreground Extraction Using Iterated Graph Cuts," ACM Transactions on Graphics (TOG) TOG, Volume 23 Issue 3, Pages 309-314, August 2004.

[13] Ravindra S. Hegadi, Basavaraj A Goudannavar, "Interactive Segmentation of Medical Images Using GrabCut", *IJMI,* Volume: 3, Issue: 3, Pages: 168-171, 2011.

[14] Y. Wang, et al., "Compressive background modeling for foreground extraction," *Journal of Electrical and Computer Engineering,* vol. 2015, pp. 1–8, 2015.

[15] Bouwmans, "Recent Advanced Statistical Background Modeling for Foreground Detection: A Systematic Survey. Recent Patents on Computer Science," *RPCS 2011,* 4(3):147–176, September 2011.

[16] T. Bouwmans, "Traditional and recent approaches in background modeling for foreground detection: An overview," *Computer Science Review,* 11(31-66), May 2014.

[17] Yubing Li, J. Z. "Grab Cut Image Segmentation Based on Image Region," *International, Conference on Image, Vision and Computing (ICIVC),* June 2018.

[18] M. Moussa, et al., "Comparative study of statistical background modeling and subtraction," *Indonesian Journal of Electrical Engineering and Computer Science,* vol/issue: 8(2), pp. 287–295, 2017.

[19] Boren Li and Mao Pan, "An Improved Segmentation of High Spatial Resolution Remote Sensing Image using Marker-based Watershed Algorithm," *2012 20th International Conference on Geoinformatics,* IEEE, Hong Kong, pp. 1-5. pp. 98-104, 2012.

[20] Vacavant, A.; Chateau, T.; Wilhelm, A.; Lequievre, L., "A Benchmark Dataset for Outdoor Foreground/Background Extraction," In *Proceedings of the 11th Asian Conference on Computer Vision,* Daejeon, Korea, pp. 291–300, 5–9 November 2012.

[21] Bashir, F.; Porikli, F., "Performance Evaluation of Object Detection and Tracking Systems," In *Proceedings of the 9th IEEE International Workshop on Performance Evaluation of Tracking Surveillance,* New York, NY, USA, 18 June 2006.

[22] Open CV Open Source Computer Vision. (n.d.). Retrieved from Interactive Foreground Extraction using Grab Cut Algorithm: https://docs.opencv.org/3.1.0/d8/d83/tutorial_py_grabcut.html.
[23] Open CV Open Source Computer Vision. (n.d.). Retrieved from Contours: https://docs.opencv.org/3.3.1/d4/d73/tutorial_py_contours_begin.html.
[24] OpenCV Open Source Computer Vision. (n.d.). Retrieved from Image Segmentation with watershed algorithm: https://docs.opencv.org/3.1.0/d3/db4/tutorial_py_watershed.html.
[25] OpenCV Open Source Computer Vision. (n.d.). Retrieved from Face Detection Using Haar- Cascades: https://docs.opencv.org/3.3.0/d7/d8b/tutorial_py_face_detection.html.