Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 165 (2019) 336–342

Procedia Computer Science

www.elsevier.com/locate/procedia

INTERNATIONAL CONFERENCE ON RECENT TRENDS IN ADVANCED COMPUTING 2019, ICRTAC 2019

Image Segmentation in Constrained IoT Servers

Nishant Sharma*, Parveen Sultana H, Rahul Singh, Shriniwas Patil, Sumit Pareek

Vellore Institute of Technology, Vellore and 632014, India

Abstract

Image segmentation forms an important concept in the computer vision technology. Image segmentation breaks the image into boundaries that differentiate meaningful components. For computer vision to realize its full potential it is essential that the image segmentation algorithms give accurate results in a fast and efficient way. In hierarchical architecture based IoT networks set up to "see" the world, methods of computer vision need more analysis. The need for low cost setup for IoT networks in terms of memory and their computational capabilities demands research for developing methods that are resource sensitive and can be successfully integrated into such networks of low end IoT servers. Addressing this need, a refined graph cut segmentation technique for low to medium resolution images and for constrained devices is presented in the paper. Implementation and analysis of the refined graph cut segmenter for linux based IoT servers is discussed. A comparison with the contemporary segmentation methods under similar constraints is also presented.

Keywords: Foreground extraction; Graph-cut segmentation; IoT; Image processing; Min-cut max-flow; RGCS; Segmentation

1. Introduction

Computer vision is a branch of computer science that is aimed at making the computers better at gaining higher

* Corresponding author:. Tel.:+91-8894645564.
E-mail address: nishant.sharma2018@vitstudent.ac.in

level understanding from digital images and videos. Image segmentation is an important process in computer vision of breaking down an image into sets of pixels that can differentiate the boundary of one object in the image from that of the other. In this paper, the graph cut segmentation technique is discussed and a refined graph cut segmenter is proposed with respect to Internet of Things based computer vision systems. A comparative analysis with contemporary segmentation techniques is also discussed in a later section.Internet of things (IoT) is an ever-growing concept of connecting all real world things in a giant network, and is increasingly becoming a part of human life. It finds application and commands presence in even the remotest areas of the developing nations. In hierarchical architectures in IoT networks, a relatively more powerful Linux based machine acts like a cluster head to a network of constrained IoT end nodes. These cluster heads are not computational heavy weights and are constrained in resources, albeit relatively less than the end nodes in the network. Making these cluster heads and end nodes capable of processing to their full potential within these constraints makes for a load balanced overall hierarchical structure reaching all the way up to the root server. These hierarchical architectures benefit from using efficient techniques based on the requirements of the application and the constraints on their hardware. IoT networks designed to process on images (like humans), and to take action based on the results of the processed image will require image segmentation methods employed by them to be extremely 'good' (i.e. accurate and efficient). EECS Berkeley has a benchmark for various boundary detection algorithms against human accuracy represented as score between 0 and 1 [12]. These segmentation results can then be used by some object detection engine to extract the features of an object. These features can identify an object (as is known in the real world) with the help of some precompiled database. A reasoning process may interact with thisdatabase and may take appropriate action based on the result of the object identification, much like a human being. These action taking systems, for instance, could alert the authorities of critical situations in the areas of video surveillance and remote sensing, and can really assist humans in making a safer society to live in. In the hierarchical architecture consisting of a root server and the corresponding sub layer servers, segmentation techniques introduced into the sub layer servers can do the essential load balancing for the root server. In varying degrees of constrained servers, techniques suitable for the hardware and computational capacity may be chosen. The constrained sub layer servers may lack a reasoner but they can assist the root server (with a reasoner) by doing load sharing in the object detection process.

2. Related Work

Review of image segmentation techniques has been done in [24], [15], [27], [16], [18], and [14]. The authors of [13] present an efficient graph based image segmentation technique. Grab cut technique and its principles are discussed in [23]. An implementation of watershed technique is presented in [17]. One of the advanced image segmentation techniques based on iterated conditional models is presented in [10]. A review of IoT based applications in done by the authors of [19]. Challenges of IoT in Indian perspective are discussed in [25]. A paper describing the benefits of hierarchical computing in IoT is discussed by the author of [26].

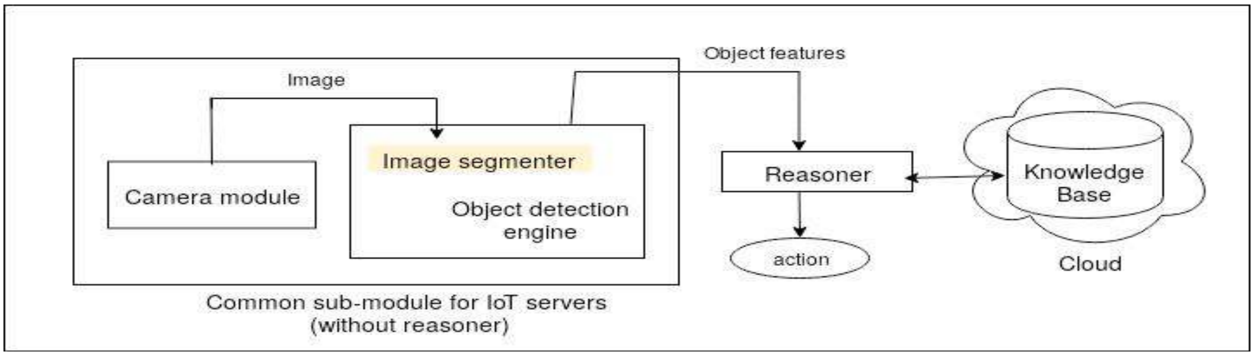


Fig. 1. System architecture for computer vision in IoT server.

3. System Architecture

Fig. 1. shows an abstract computer vision architecture for an IoT server. The discussion on image segmentation in this paper is centred on this architecture.A common sub module for IoT servers in a network is shown in the figure. An image is captured from the camera module and is passed to the object detection engine whose final objective is to extract the object features.An image segmenter forms a part of this engine and is utilized while separating the object boundaries in the image. Multiple sub layer servers (in a hierarchical network design) can provide the object detection features to the root server for various surrounding objects and the root server can reason on them. The reasoner (as part of the root server) interacts with a knowledge base that assists in the object identification. The knowledge base may be stored on the cloud owing to its large size. The reasoner may take a suitable action based on the result of thisobject identification processand the requirements of the application. An example of this action may be to alert the authorities after successfully reasoning about a security breach in a protected area based on one or more captured images of an intrusion. This paper limits itself to the image segmentation part of the object detection engine and proposes a refined graph cut segmentation technique that is sensitive to the IoT servers' computational capabilities while also giving accurate results.

4. RGCS for Constrained IoT Servers

4.1. Graph cut segmentation algorithm

The idea of the graph cut segmentation algorithm is discussed by the author of [22]. Its brief summarization is as follows. Consider Fig. 2. The image is visualized as a directed graph. This directed graph for the image consists of two types of nodes, those that lie on the foreground and those that lie on the background. Two extra nodes denoting the source and the sink are assumed. There are three types of edges for a node in the image graph. A node is connected with an edge from the source, an edge to the sink and an edge to its neighbouring nodes. A 4-neighborhood is considered, i.e. a node is attached to the node on its left, right, up and down. Source and sink have edges to (and from) a node that denote its belonging to the foreground and background respectively. The basic idea is that a node which has a higher probability of being a foreground will have an edge with a greater weight from the source, and a node that is probably background will have an edge with a lower weight from the source. This implies that a higher weight edge from the source for a foreground node will mean a lower weight edge from the node to the sink and vice versa. Nodes also connect with some edge weight to their 4 neighbourhood adjacent. A probability distribution model is needed that determines the edge weights between the source and the node, between the node and the sink, and between the node and the 4-neighborhood adjacent. The mathematical model is restated in (1).To determine the relative edge weights of the neighbouring nodes a variable (κ) has been used.In [22], the author has used a naive Bayesian classifier on the RGB values of the image nodes to determine the probability of a node in the image belonging to either the foreground or the background. A Bayesian classifier is trained on the training data fed to it from the image itself and is based on the metric of calculated mean RGB value and variance from the training data. Assuminga Gaussian multivariate normal distribution [11] a classifier is constructed which is used to assign the respective foreground and background probabilities for each node in the image given its RGB vector([RGB])w.r.t. the obtained mean and variance (or standard deviation)results from the training data. In (1), σ is the standard deviation obtained from the training data.

$$\begin{aligned}
edge\_weight_{source \to node} &= \frac{P_{foreground}([RGB]_{node})}{P_{foreground}([RGB]_{node}) + P_{background}([RGB]_{node})} \\
edge\_weight_{node \to \sin k} &= \frac{P_{background}([RGB]_{node})}{P_{foreground}([RGB]_{node}) + P_{background}([RGB]_{node})} \qquad (1) \\
edge\_weight_{node_1 \to node_2} &= \kappa e^{\frac{-\|[RGB]_{node1} - [RGB]_{node2}\|^2}{\sigma}}
\end{aligned}$$

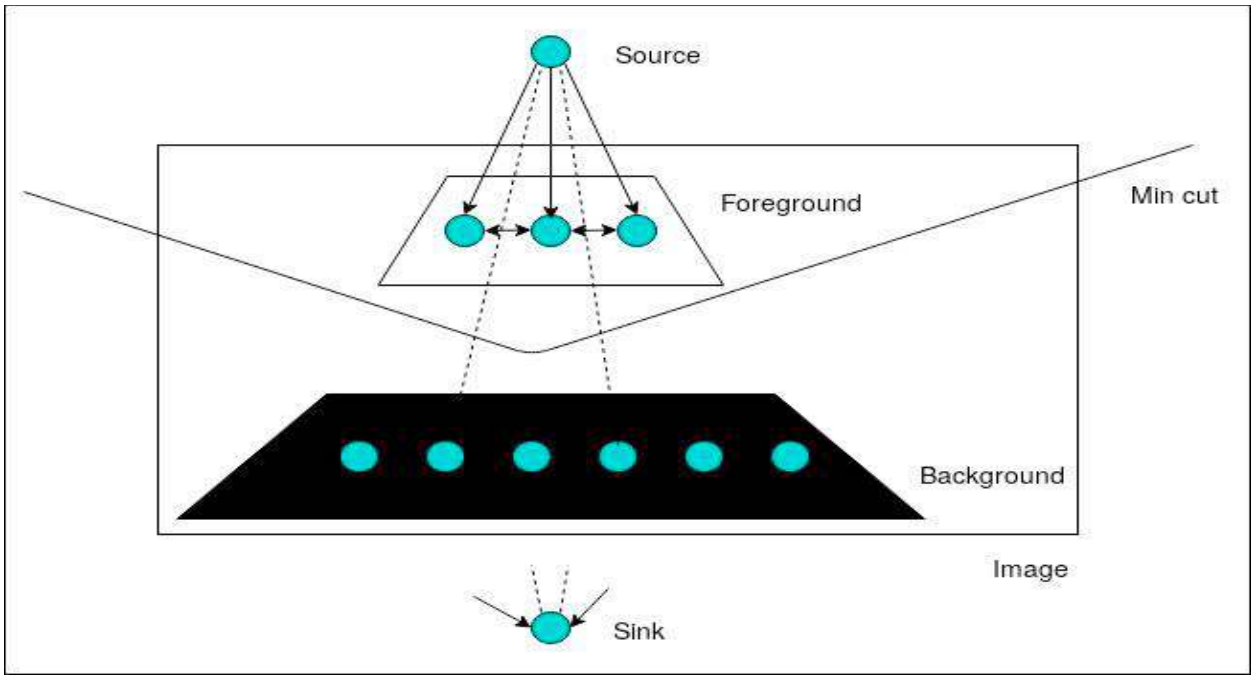


Fig. 2. Foreground extraction using graph cut segmentation technique.

Once the graph is constructed, foreground extraction is performed using a standard min-cut max-flow algorithm which cuts through the background edges from the source to various nodes as shown in Fig.2.

4.2. RGCS (Refined Graph Cut Segmenter)

The methodology used by the author of [22], although extremely low taxing on the resources and an ideal base model for low end IoT nodes, is not contemporary. Firstly, the algorithm is serial in nature. It does not take advantage of the multiprocessing abilities now common to most relatively powerful Linux based IoT servers. For example, raspberry pi (a small and affordable IoT server) [28] has a quad-core CPU. There are certain calculations that can be parallelized. Among these is the process(for all nodes) of assigning edge weights in accordance with (1) for each edgebetween the node and the source, and the node and the sink. The mathematical calculations on column mean and variances for RGB values that are required to train the Bayesian classifier may also be performed utilizing the parallel processing capabilities of the machine. The Process and the (synchronized) Queues methods of the multiprocessing library of python [6] are utilized for parts of the program that could benefit from parallel processing and where exchange of results between processes is required, such as in case of a queue that stores the mean of a large column of color values and must return it to the caller before exiting. On other parts of the program where simultaneous work could be done but no exchange of data needed to happen like the edge weight calculation of individual nodes with source and sink, the threading library of the python [9] may be utilized. Although raspbian os on raspberry pi 3 supports a 32 bit armhf architecture, 64 bit variants of pi and suitable complementing os are readily available. On a 64 bit machine using a debian/linux based os, efficiency and speed can be further enhanced using such tools as Graph tools. Graph tools [2] is a C++ based python module that provides means for graph and algorithms' visualization and analysis, and that uses OpenMP [8] to provide fast implementations of standard graph algorithms such as the min-cut max-flow algorithm by enabling high performance computing and parallel programming. In contrast to the Edmonds-Karp min-cut max-flow algorithm used in [22] from the python-pygraph minmax-module [5], Graph tools provides more efficient OpenMP based min-cut max-flow implementations of which boykov_kolmogorov_max_flow() is used in RGCS.

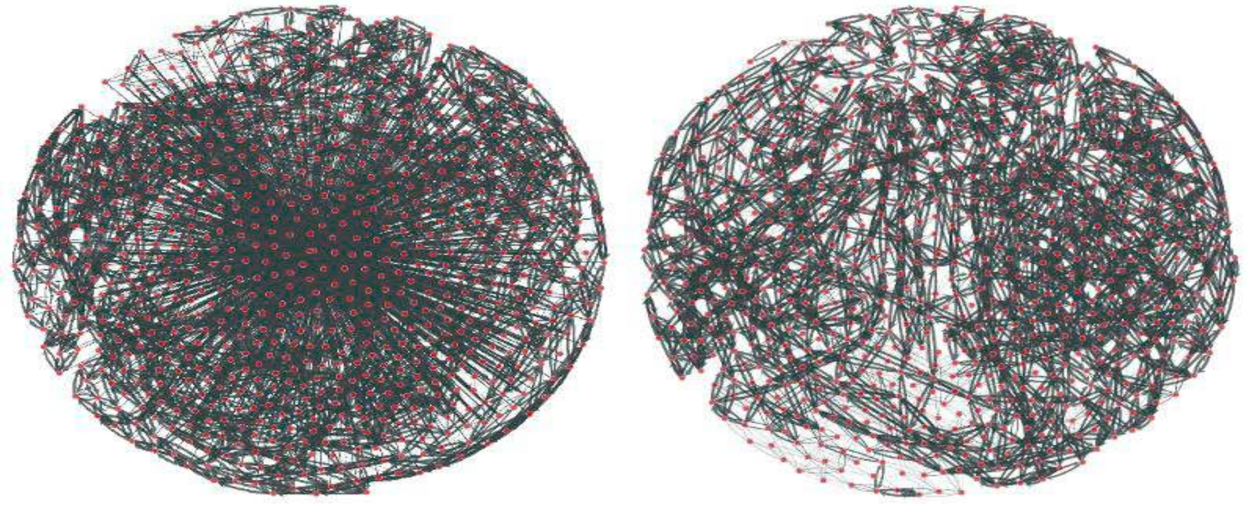


Fig. 3. (from left) (first) Graph for the test image with edge weights as in (1); (second) Residual capacity result graph (foreground extraction).

5. Raspberry Pi as the IoT server

A raspberry pi client is set up on Ubuntu 18.04 (bionic beaver) for simulation. Raspbian OS is installed and configured on pi. Raspbian is based on debian buster and supports 32 bit armhf architecture. Networking is then configured on pi. A camera module is set up on pi as demonstrated in [1]. An image is taken from the camera module and is usedas a test image for experimentation.

6. Results and Discussion

6.1. Foreground extraction by RGCS

The test image is resized to 400 × 265 px. The segmentation results for the min-cut max-flow boykov_kolmogorov_max_flow() of Graph tools used in the experiment are visualized in Fig.3. Fig.3. (first) depicts the graph for the test image constructed by the graph tools visualizer where each edge represents either an edge between the source and a node, or a node and the sink, or a node and its 4 neighbours, with edge weights calculated from (1). The edges in Fig.3. (second) form the extracted foreground after the application of the min-cut max-flow algorithm. It is obvious from the extraction shown in Fig.4. (first) that a significant part of the test image is foreground. As a result Fig.3. (second) has a dense foreground extraction.

6.2. Comparison with the contemporary watershed and grabcut segmentation techniques

In watershed segmentation technique an image is considered similar to a geographical area containing either the sky touching mountains or the valley areas between those mountains [20], and is a very efficient technique for segmentation. Grab cut discussed comprehensively in [21] on the other hand is a foreground extraction method extending the graph cut segmentation principles (similar to the basic principles of the RGCS) and makes it an ideal candidate for comparative analysis. OpenCV [7], which is an open source library for the implementation of computer vision and machine learning, provides an implementation for the grabcut [4]andthe watershed algorithm [3]. OpenCV grabcut notably iterates on segmentation outputs until the results converge. Watershed algorithm, similarly, is prone to over-segmentation. An instance for the simulation run on the test image for RGCS, the grabcut and the watershed algorithm is given in Fig. 4. Some of the benchmarks for the four algorithms: RGCS (without Graph tools), RGCS (with Graph tools), OpenCV grabcut and OpenCV watershed are given in Table 1.

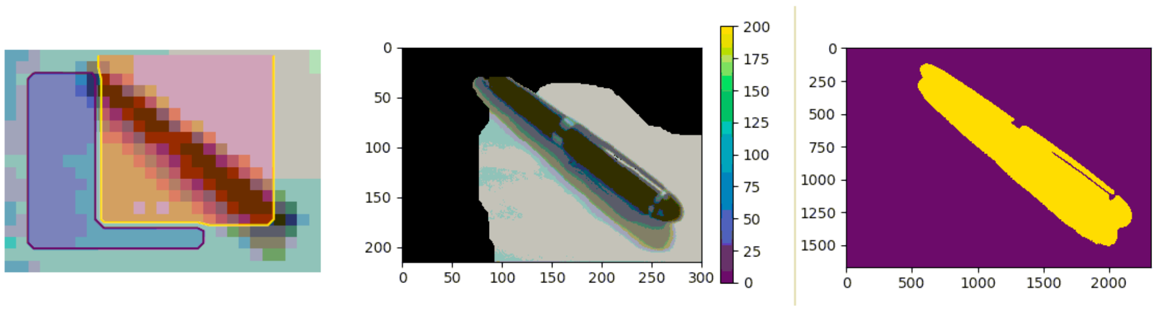


Fig. 4. (from left) (first) RGCS; (second) OpenCV grabcut; (third) OpenCV watershed.

Table 1. Benchmarks for the algorithms.

Algorithm	Runtime	Image resolution	Computational complexity	Storage requirements (Modules and Packages)
RGCS (without Graph tools)	Slow	Low, medium	Low	Low
RGCS (with Graph tools)	Comparable to OpenCV grabcut	Low, medium	Low	Medium
OpenCV grabcut	Fast	Low, medium	Medium	Medium
OpenCV watershed	Slower than RGCS (with Graph tools)	Low, medium, high	Medium	Medium

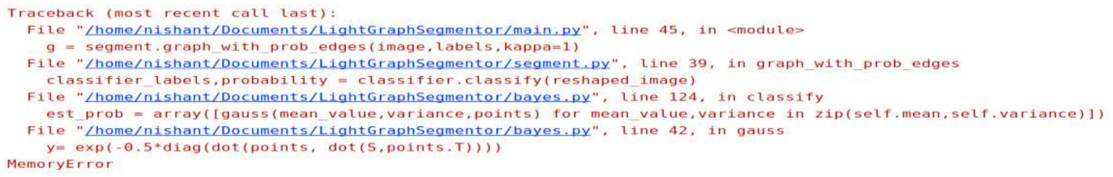
RGCS improved under the refinements provides an equivalent performance for a test image when compared to the OpenCV's grabcut and watershed implementations. For low and medium resolution images RGCS (with Graph tools) and the OpenCV grabcut give comparable performance on a linux based 64 bit IoT server. The OpenCV watershed takes longer time for low and medium resolution images but provides results when dealing with high resolution images.

7. Conclusion

Thepaper discusses IoT applications that are set up to reason on digital images, and the importance of image segmentation techniques in realizing computer vision in its full potential. It discusses a refined graph cut segmenter (RGCS) that does foreground extraction based on the probability that a node in an image graph belongs to either the foreground or the background. The refinements to the basic principle involve using the parallel processing environment now common to IoT servers. The underlying graph representation and the min-cut max-flow algorithm may also be made more efficient using state of the art and parallel programming based graph algorithm libraries. RGCS is well suited for segmenting low and medium resolution images in an IoT server. The issue of memory error in images of high resolution for the RGCS is outlined in the Appendix A. Future work may involve addressing this issue. Based on the constraints on server resources and the requirements of the application, an appropriate segmentation algorithm may be chosen based on the benchmarks given in Table 1.

Appendix A. Images of high resolution

When passing the original test image taken from the camera module of resolution 2584×1704 px to RGCS, the calculations outreach the memory bounds leading to a memory error as shown in Fig. 5.



```
Traceback (most recent call last):
  File "/home/nishant/Documents/LightGraphSegmentor/main.py", line 45, in <module>
    g = segment.graph_with_prob_edges(image,labels,kappa=1)
  File "/home/nishant/Documents/LightGraphSegmentor/segment.py", line 39, in graph_with_prob_edges
    classifier_labels,probability = classifier.classify(reshaped_image)
  File "/home/nishant/Documents/LightGraphSegmentor/bayes.py", line 124, in classify
    est_prob = array([gauss(mean_value,variance,points) for mean_value,variance in zip(self.mean,self.variance)])
  File "/home/nishant/Documents/LightGraphSegmentor/bayes.py", line 42, in gauss
    y= exp(-0.5*diag(dot(points, dot(S,points.T))))
MemoryError
```

Fig. 5. RGCS-Memory error for an image of high resolution.

References

[1] Camera configuration [internet]. Available from: https://www.raspberrypi.org/documentation/configuration/camera.md.

[2] graph tools - efficient network analysis [internet]. Available from: https://graph-tool.skewed.de/.

[3] Image segmentation with watershed algorithm [internet]. Available from: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_watershed/py_watershed.html.

[4] Interactive foreground extraction using grabcut algorithm [internet]. Available from: https://docs.opencv.org/3.4.3/d8/d83/tutorial_py_grabcut.html.

[5] Module minmax [internet]. Available from: http://www.chiark.greenend.org.uk/doc/python-pygraph/docs/pygraph.algorithms.minmax-module.html.

[6] multiprocessing process-based threading interface [internet]. Available from: https://docs.python.org/2/library/multiprocessing.html.

[7] Opencv [internet]. Available from: https://opencv.org/.

[8] OpenMP [internet]. Available from: https://www.openmp.org/.

[9] threading thread-based parallelism [internet]. Available from: https://docs.python.org/3/library/threading.html.

[10] Besag, Julian. (1986) "On the statistical analysis of dirty pictures." *Journal of the Royal Statistical Society: Series B (Methodological)* **48:** 259–279.

[11] Do, Chuong B. (2008) More on multivariate gaussians [internet]. Available from: http://cs229. stanford. edu/section/more on gaussians. pdf.

[12] Boundary detection benchmark: Algorithm ranking [internet]. Available from: https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/bench/html/algorithms.html.

[13] Felzenszwalb, P.F., and D.P. Huttenlocher. (2004) "Efficient graph-based image segmentation." *International journal of computer vision* **59**: 167–181.

[14] Fu, King-Sun, and J. K. Mui. (1981) "A survey on image segmentation." *Pattern recognition* **13**: 3–16.

[15] Gonzalez, Rafael C., Richards E. Woods, and Steven L. Eddins. (2009). *Digital image processing using matlab*, Gatesmark publishing.

[16] Haralick, Robert M., and Linda G. Shapiro. (1985) "Image segmentation techniques." *Computer vision, graphics, and image processing* **29**: 100–132.

[17] Kaur, Amandeep. (2014) "Aayushi,image segmentation using watershed transform." *International Journal of Soft Computing and Engineering (IJDCE)* **4**: 5–8.

[18] Kumar, Vinod, Tauj Lal, Piyush Dhuliya, and Diwaker Pant. (2016) "A study and comparison of different image segmentation algorithms." *International Conference on Advances in Computing, Communication, & Automation (ICACCA) (Fall): IEEE* **2**: 1–6.

[19] Nagakannan, M., C. Johnson Inbaraj, K. Mukesh Kannan, and S. Ramkumar. (2018) "A recent review on iot based techniques and applications." *International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud): IEEE* **2**: 70–75.

[20] Preim, Bernhard, and Charl P. Botha.(2013). *Visual Computing for Medicine*, 2 ed., Author.

[21] Rother, Carsten, Vladimir Kolmogorov, and Andrew Blake. (2004) "Grabcut: Interactive foreground extraction using iterated graph cuts." *ACM transactions on graphics (TOG)*: 309–314.

[22] Solem, Jan Erik. (2012). *Programming Computer Vision with Python: Tools and algorithms for analyzing images*, O'Reilly Media Inc.

[23] Talbot, Justin F., and Xiaoqian Xu. (2006) "Implementing grabcut." *Brigham Young University* **3**.

[24] Xess, Monika, and S. Akila Agnes. (2014) "Analysis of image segmentation methods based on performance evaluation parameters." *Int. J. Comput. Eng. Res* **4**: 68–75.

[25] Yadav, Er Pooja, Er Ankur Mittal, and Hemant Yadav. (2018) "Iot: Challenges and issues in indian perspective." *International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU): IEEE* **3**: 1–5.

[26] Yang, Zhihe. (2017) "Hierarchical computing: A high performance computing architecture for data-processing in iot era." *International Conference on Systems and Informatics (ICSAI): IEEE* **4**: 1698–1702.

[27] Yuheng, Song, and Yan Hao. (2017) "Image segmentation algorithms overview." *arXiv preprint arXiv:1707.02051.*

[28] Raspberry Pi Blog [internet]. Available from: https://www.raspberrypi.org/blog/.