Università della Svizzera italiana

Faculty of Informatics

USI Technical Report Series in Informatics

A High Performance Video Segmentation Framework

Liudmila Karagyaur[1], Lorenzo Ferri[1], Vanessa Braglia[1]

[1] Faculty of Informatics, Università della Svizzera italiana, Switzerland

Abstract

Edge detection is a ubiquitous technique in scientific computing. It is widely utilized in content based video retrieval for the search of digital information in large datasets, and also plays a critical role in the development of self-driving vehicles that need to perform real-time feature detection. We present a high performance video segmentation framework that is able to find the main features of a video, and to follow their trajectories. Initially, we evaluate the performance and solution quality of k-means clustering algorithms, in order to ensure that it is a suitable choice for the application at hand. Subsequently, a convolutional neural network is trained to label the obtained clusters, thus enabling the extraction of the main features of the video. The process is executed in parallel for each frame of the video, in order to reduce the overall runtime of the routine.

Report Info

Published
June 2019

Number
USI-INF-TR-2010-4

Institution
Faculty of Informatics
Università della Svizzera italiana
Lugano, Switzerland

Online Access
www.inf.usi.ch/techreports

1 Introduction

Nowadays edge detection techniques have several applications. A content based video retrieval is widely required for searching digital information in large databases, in order to improve text based retrieval systems [1]. It also has an important role in the development of self-driving vehicles, that need to perform a real-time feature detection [2], and in the field of medicine, in the analysis of digital images of pathological conditions, such as tumors [3].

Extracting the main features of a video is a crucial step in the process of image recognition. It facilitates procedures such as pattern detection and classification [4], while extracting complete and clear features from images can still be a challenging task in computer vision [5].

The focus of this work is to find the main features of a video and to be able to follow their trajectories. In particular, we will analyze single video frames using edge detection, which is an image segmentation technique.

There exist a plethora of clustering algorithms that perform efficient image segmentation, which is a fundamental step in feature extraction [6]. Clustering algorithms used for this purpose are k-means, that can be improved using a subtractive clustering algorithm, which generates the initial centroid based on the potential value of the data points [7], but also a DP clustering algorithm that can directly give the number of clusters based on graph generated by the image [8], and non parametric clustering algorithms [9].

In this work we present an efficient approach to the video segmentation problem. It is focused on the selection of fast, inexpensive techniques that, when combined, lead to remarkable results comparable to those generated by more sophisticated algorithms.

The organization of the rest of the paper is illustrated in Figure 1. The first task to be completed is the application of the k-means clustering algorithm to the video, previously subdivided into frames. The goal is to be able to identify the main features among other secondary objects and obtain isolated clusters for each of them.

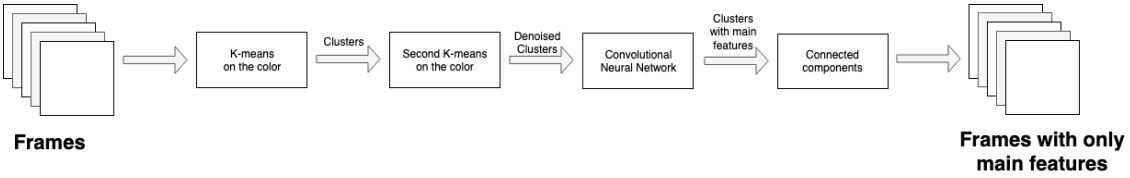

Figure 1: Workflow: extraction of the main components from the original frames.

The next step consists in being able to distinguish between the clusters that contain the main items of the video and those that enclose noisy components. In order to do that, we are going to use a convolutional neural network that is going to label all the generated clusters, identifying the ones that will be processed in the later stages.

Each cluster classified as a main component of the frame is then exposed to an algorithm that identifies all the connected components. The goal we want to reach is the partition of the cluster into its individual items.

The final step consists in reducing the overall runtime of the routine, a task that is accomplished by executing the whole process in parallel for a suitable number of frames, depending on the number of processors available in the machine. The characteristics and the mathematical details behind the *k*-means algorithm, convolutional neural network, connected components, and multiprocessing are contained in section 2, while the results obtained by means of the application of previously listed methods are discussed in section 3.

2 Algorithms and Mathematical Background

In order to apply the algorithms and adapt them to a specific problem it is fundamental to fully understand all the theoretical and mathematical concepts characterizing their process. For this reason, before proceeding with the discussion and visualisation of the video segmentation framework results, in this section is presented the theoretical background of the methods used.

2.1 k-means Clustering Algorithm

In image segmentation, *k*-means is one of the most popular clustering algorithms. It is intuitive, easy to implement, and converges monotonically. In spite of this, *k*-means also presents some drawbacks such as the fact that the number of clusters has to be known a priori and different initial seeds can give rise to different outputs, because of the randomness of the initial component [10].

The algorithm uses an iterative approach, as represented in Figure 2 and formalized in Algorithm 1. Starting from an initial set of random centroids, at each iteration it generates the clusters by minimizing the distance between each point and the centroids, based on a given proximity measure. Convergence is achieved when points are no longer interchanged between the clusters.

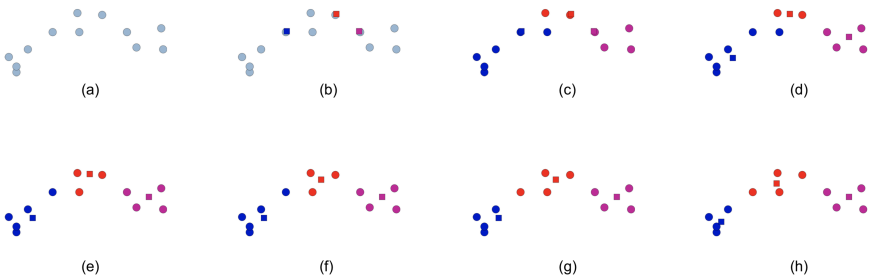

Figure 2: iterations of *k*-means clustering algorithm applied to a group of datapoints (a) from initial random centroids (b) to final clusters (h). We can observe how the center of mass is moving around, the points are assigned to clusters, and the centroids are updated in (c)–(g).

Algorithm 1 k-means clustering.

```
procedure k-MEANS
    Choose K random points as initial centroids
    while not converged do
        Assemble K clusters by minimizing the distance between each point and centroids
        Compute new centroids
    end while
end procedure
```

The ultimate goal of the algorithm is to partition N initial data points into k clusters. Given the dataset $\{x_1, x_2, \ldots, x_N\}$ we want to obtain a previously provided number k of clusters denoted $\{C_1, C_2, \ldots, C_K\}$. The objective function usually employed by k-means is the sum of squared errors, denoted as SSE. The aim is to minimize SSE for a given set of centroids c_k [11].

k-means can be formalized mathematically as follows:

$$SSE(C) = \sum_{k=1}^{K} \sum_{x_i \in C_k} \|x_i - c_k\|^2,$$

$$c_k = \frac{\sum_{x_i \in C_k} x_i}{|C_k|}.$$

Different proximity measures can be used when the k-means algorithm is applied. The most popular one is the Euclidean distance metric. However in the particular case of this application the distance between the pixels is computed considering their color in RGB format. This means that the data point x_i will be represented by a three-dimensional array, whose elements will correspond to the R, G, and B components of the color.

2.2 Connected Components

Connected components is a viable alternative to more common clustering algorithms, like k-means. In order to understand how the algorithm works it is necessary to introduce some concepts of graph theory.

A graph $G = (V, E)$ is a set of elements $v \in V$ called vertices or nodes and a set of $e \in E$, the edges connecting the nodes. A path is a sequence of edges and vertices where each subsequent vertices are adjacent to each other and neither edges nor vertices are repeated [12]. A connected component of an undirected graph (i.e. all edges are bidirectional) is a subgraph where each two nodes are joined together by a path and at the same time there is no connection with other nodes in the graph. Connected components is an algorithm that applies concepts of graph theory in order to identify those sets of nodes that are considered similar based on a given heuristic. In particular, two nodes are considered part of the same connected component if there is a path between them, as represented in Figure 3.

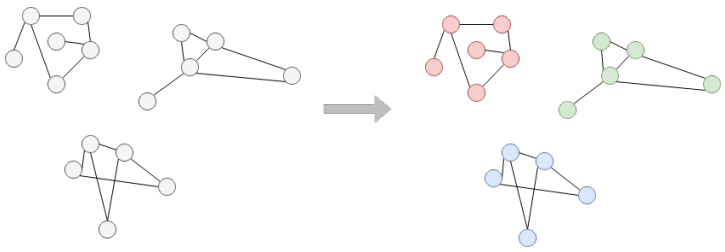

Figure 3: An example of identification of 3 connected components in a graph with the connected components algorithm.

Identifying connected components could be a very useful tool if we want to isolate clusters in a set of items. The application of this algorithm on images will consider each pixel as one node of the graph. The metric used to evaluate the connection between two nodes is based on the color of the pixels. A weighted

alternative of the algorithm can also be considered. It recognizes two pixels in the same connected component if their distance in terms of colosr is above a certain threshold.

2.3 Convolutional Neural Network

A convolutional neural network is a deep learning algorithm that is usually applied in image processing [13]. Over the years, using convolutional neural networks has become the standard approach when it comes to image classification because of their speed and accuracy. Like other neural network models, it consists of an input layer and an output layer, with several hidden layers in between. The hidden layers of a convolutional neural network are usually convolutional layers, max-pooling layers, followed by some common fully connected layers.

The goal of the convolutional layer is to reduce the input size while maintaining the spatial relations of the pixels in the image. A convolutional layer takes as input a tensor, usually of shape batch size × channels × height × width, and a kernel. The number of channels depends on the characteristics of the image: usually, taking into consideration the color, there are three channels that correspond to the three RGB values. On the other side, kernels define the output size of the layer and how the input image is processed. They can have different sizes and apply different functions to the pixels, depending on how they are defined.

For example as shown in Figure 4, considering a 3 × 3 kernel, a mask of the same dimension scans over the whole input image, pixel by pixel, computing the value in the centre with a weighted sum of all the pixels in the mask. The weights are defined by the selected kernel.

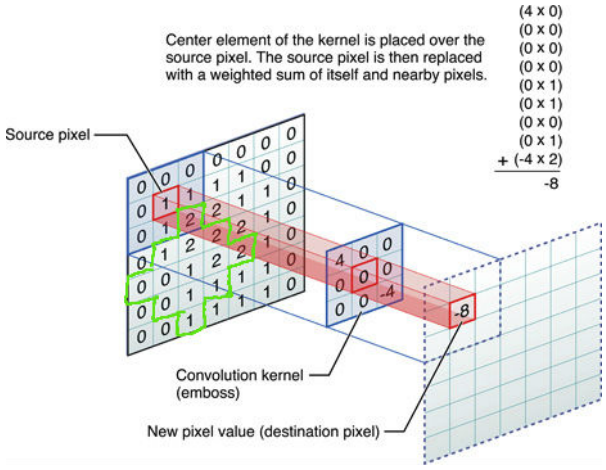

Figure 4: A visual example of a convolutional layer with a 3×3 kernel [14].

After the application of the convolutional layer, the goal of the pooling layer is to further reduce the input size. However, it is a lossy layer, meaning that it will not conserve all the information given by the input. It usually divides the image in many 2×2 groups of pixels and substitutes for each of them a single pixel, the one with the maximum value. This halves the size of the image, as can be seen in Figure 5.

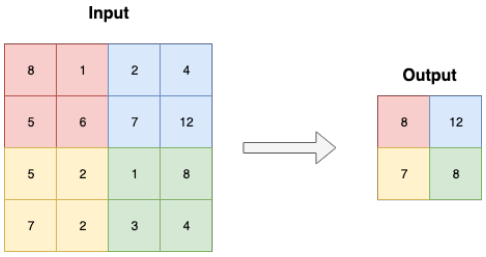

Figure 5: A visual representation of a max-pooling layer.

The final step that leads to the classification is performed by a certain number of fully connected layers. As the name suggests, they are layers where each neuron is connected to all the neurons of the previous and

the next layer. Each neuron carries a weight that is multiplied by the input, which is determined by the sum of all the incoming nodes. It results in an affine transformation of the input X that can be mathematically formalized as

$$Y = X * W + b,$$

where Y is the output, W is the matrix containing the neuron weights, and b is a constant value used to shift the result of the computation, called bias.

2.4 Multiprocessor Parallelization

Multiprocessors refers to the use of two or more CPUs (central processing units) contained in a single computer system that supports the execution of threads collaborating on a single task. The coordination and the usage of the single processors is controlled by the same operating system and all the processors share the same main memory by means of a shared address space.

The usage of multiple processors instead of one presents several advantages. One of the most important ones is the increase of the throughput, which means that more work can be executed and completed in a unit time. However, an important drawback could be the cost of the communication among different processors. For this reason the parallelization of a program with multiprocessing is particularly convenient if the tasks are independent of each other.

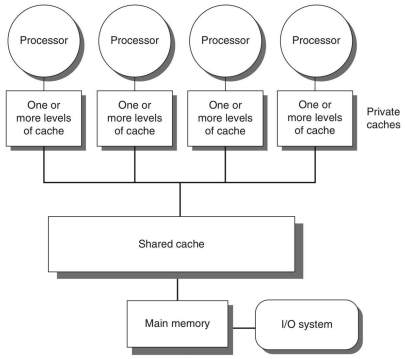

Figure 6: Structure of a centralized shared-memory multiprocessor [15].

3 Results and Discussion

This section presents, step by step, the results of the high performance video segmentation framework, obtained by applying the methods previously discussed to a specific case. The way the algorithms are adapted and manipulated in order to fit the test case is also examined.

3.1 Test Case

The specific video we are considering represents a clip of a billiard game with a duration of about 2 minutes, filmed with a static camera. The video subdivided into frames generates 3582 images, some examples of which are shown in Figure 7. Each frame has dimension of 1280×720, for a total of 921,600 pixels.

Figure 7: Examples of frames extracted from 5 different moments of the footage.

In the video there are two teams represented by red and yellow balls; the aim of the application is to separate and identify the members of each team. The final goal is to reassemble the video with only its

main features, represented by the single balls and display a counter to keep track of each team's members left on the table.

3.2 Video Feature Extraction

We hereby suggest an effective approach to perform video feature extraction. All the methods described are applied to all the single frames of the clip, but for the sake of brevity only one example will be presented.

Once the frames are extracted from the video by means of the OpenCV library for Python, the k-means algorithm is applied to them for the first time in order to separate the main features from those of less interest. However, before applying the clustering algorithm to the frames it is convenient to get rid of the possible influence of the secondary objects. For this purpose the background will be set to black on each frame before proceeding with the analysis.

The clustering algorithm is performed using the function `kmeans` from the OpenCV library. The main parameters given in input are the vectorized image, the number of clusters, the termination criteria, and the flag specifying how the initial centroids are chosen. The image is transformed into a number of pixels×3 array containing the RGB values; the number of clusters is set to 5 because the initial idea was to identify red balls, yellow balls, white ball, table, and background. In Figure 8 it can be observed how the algorithm divides the image into its main components based on the color; the result is not exactly as expected.

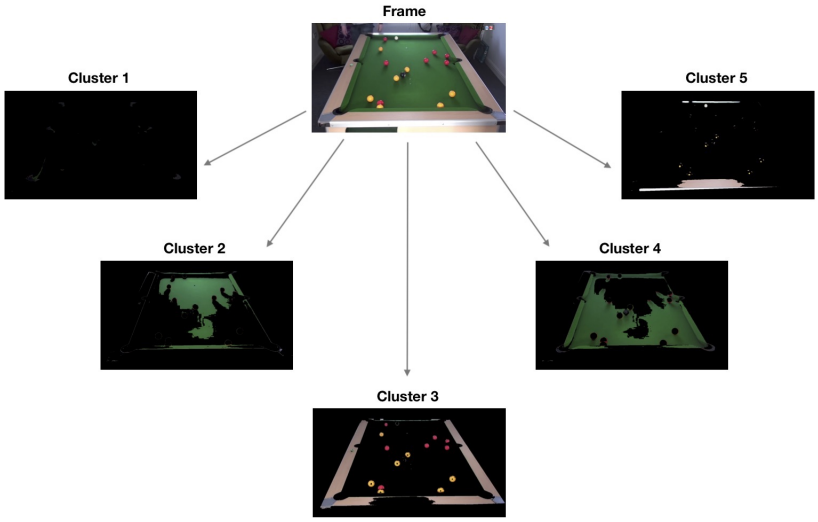

Figure 8: The clusters resulting from the first application of the k-means algorithm.

As previously stated, the result wanted from the application of the k-means algorithm was the separation of the members of each team into two different clusters. However, it is easy to notice that this task has not been accomplished in the previous step. In spite of this, we have obtained a cluster that contains the member of the two teams, i.e. cluster number 3, and a second application of the k-means algorithm can give rise to the division of the teams into two different clusters.

Since there is no way to know in which order the clusters will be generated and therefore it is impossible to identify the cluster that contains the teams, the k-means clustering will be applied for the second time to all the resulting frames from the previous step. As before, the algorithm generates 5 clusters from each input, identifying the two teams and removing the noise, as represented in Figure 9.

From the double application of the k-means algorithm we obtain 25 clusters; the next goal will be to identify which of them contain the main features, i.e., the two teams.

In order to achieve this result we have used a convolutional neural network, trained to classify a cluster as yellow team, red team, or noise. The training is performed based on a set of 500 frames classified manually. The precision achieved is over 99%.

Once the two teams have been identified and labeled by the convolutional neural network, the next goal is to identify each member of the two teams, separating them, as can be observed in Figure 10, by applying the connected components algorithm. This will allow us to keep track of the trajectory of every single member and identify the number of balls present on the billiard table.

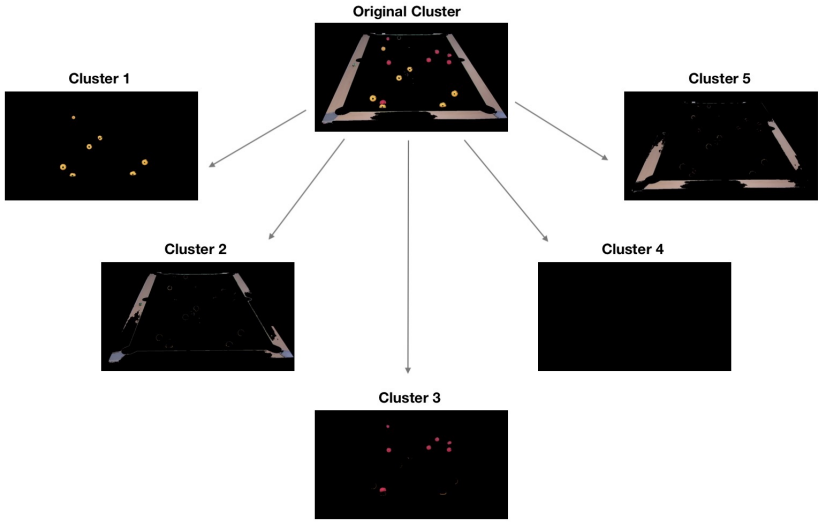

Figure 9: The clusters resulting from the second application of the k-means algorithm to Cluster 3 from Figure 8.

The connected component clustering is performed by the SciPy library that employs a Gaussian filter and a function that labels features in an array based on a threshold that in this particular case was set to 50.

Once the connected components are identified the frame is reassembled, as shown in Figure 11.

After this procedure is applied to each frame, the final video is reassembled with only the members of the two teams, displaying the balls counter, as visualized in Figure 12.

3.3 Parallelization and Scalability

All the processes explained until now, applied to a single frame, take about 94 seconds to execute on a MacBook Pro with a 2.2 GHz Intel Core i7 processor and a memory of 16 GB 2400 MHz DDR4. So, to run the feature extraction techniques on all the frames on this machine would require 336,708 seconds, i.e., almost 94 hours. For this reason, the framework has been parallelized and can be run on multiple processors, due to the independence of the frames. For these simulations, the ICS cluster that is composed of 42 nodes is used. Our application is run on regular nodes, with 2 CPUs Intel Xeon E5-2650 v3 @ 2.30 GHz, 20 cores and a RAM of 64 GB DDR4 @ 2133 MHz each.

In order to show the scalability of the video segmentation framework, it is run on 1, 2, 4, 8, and 16 nodes. It can be observed in Figure 13 that by increasing the number of nodes the execution time reduces significantly and almost ideally, i.e., doubling the number of cores for which the execution time halves. Because of the limited resources, the point of saturation, i.e., where the program has no more improvement in execution time increasing the number of nodes, could not be shown.

4 Conclusion and Future Work

The presented work is an efficient approach to the video segmentation problem. It is based on the employment of fast and inexpensive techniques that successfully identify the critical components of a video.

The main steps characterizing the high performance video segmentation framework are

(i) the double application of the k-means algorithm to the frames of a video,

(ii) the classification of the resulting clusters into three groups (i.e. yellow team, red team, and noise),

(iii) the application of the connected components clustering algorithm to identify the single members of each team,

(iv) and, finally, the reassembly of the video with only its main features (the two teams).

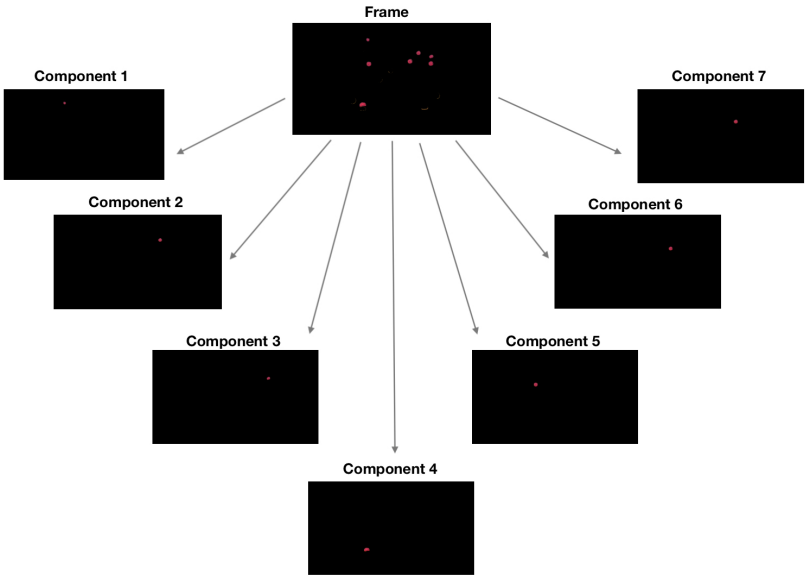

Figure 10: The clusters resulting from the application of the connected components algorithm applied to Cluster 3 of Figure 9.

Experimental results show that the method proposed leads to an efficient identification and feature extraction of the main objects in a video. The application is computationally inexpensive but could require a long execution time in the case of long videos that generate thousands of frames. This issue can be easily overcome by the fact that when the program is executed on multiple processors it has an almost ideal scalability that significantly reduces the runtime.

Future work could develop in different directions. A first option for a further evolution could be the employment of more powerful clustering algorithm such as spectral clustering, in order to improve the quality of the clusters obtained. In another possible development, the framework can be manipulated and adapted in order to fit different test cases; possible applications could be various types of sports and the identification of potential obstructive objects in autonomous vehicles.

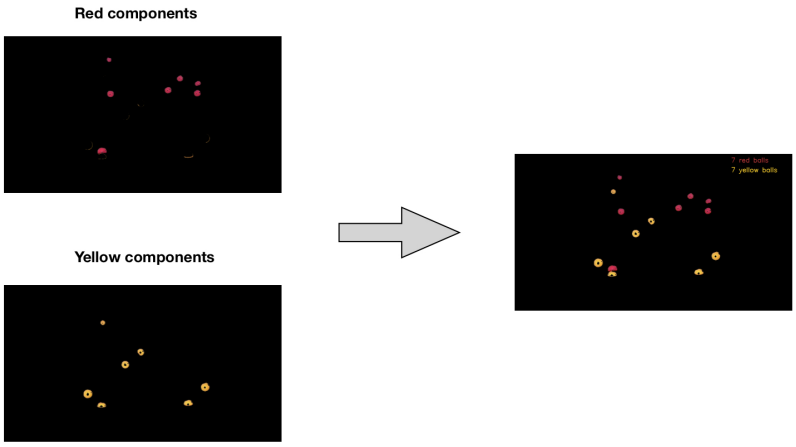

Figure 11: The reassembled frame with counter.

Figure 12: The reassembled frame with counter.

References

[1] B.V. Patel and B.B. Meshram, *Content Based Retrieval Systems*, International Journal of UbiComp (IJU), Vol.3, No.2, April 2012

[2] H. Cho, Y. Seo, B. V. K. V. Kumar and R. R. Rajkumar, *A multi-sensor fusion system for moving object detection and tracking in urban driving environments*, IEEE International Conference on Robotics and Automation (ICRA), 2014

[3] Ed-Edily Mohd, Azhari,Muhd. Mudzakkir Mohd. Hatta, Zaw Zaw Htike, Shoon Lei Win, *Tumor detection in medical imaging: a survey* , International Journal of Advanced Information Technology (IJAIT) Vol. 4, No. 1, February 2014

[4] Elnemr Heba Zayed, Nourhan Fakhreldein Mahmoud, *Feature Extraction Techniques: Fundamental Concepts and Survey*, Handbook of Research on Emerging Perspectives in Intelligent Pattern Recognition, Analysis, and Image Processing, Chapter: 13, January 2016

[5] Dong ping Tian, *A Review on Image Feature Extraction and Representation Techniques*, International Journal of Multimedia and Ubiquitous Engineering, July 2013

[6] Mark Nixon and Alberto Aguado, *Feature Extraction and Image Processing for Computer Vision* Third edition, 2012

[7] Nameirakpam Dhanachandra, Khumanthem Manglem, Yambem Jina Chanu, *Image Segmentation Using K -means Clustering Algorithm and Subtractive Clustering Algorithm* Procedia Computer Science, Volume 54, 2015

[8] Zhensong Chen, Zhiquan Qi, Fan Meng, Limeng Cui, Yong Shi, *Image Segmentation via Improving Clustering Algorithms with Density and Distance* Procedia Computer Science, Volume 55 2015,

[9] E.J Pauwels, G Frederix, *Finding Salient Regions in Images: Nonparametric Clustering for Image Segmentation and Grouping* Computer Vision and Image Understanding, Volume 75, 1999

[10] A. Oliver, X. Munoz, J. Batlle, L. Pacheco and J. Freixenet *Improving Clustering Algorithms for Image Segmentation using Contour and Region Information*, 2006 IEEE International Conference on Automation, Quality and Testing, Robotics, 2006

[11] Charu C. Aggarwal and Chandan K. Reddy *Data Clustering, Algorithms and Applications*,Chapter 4, 2014

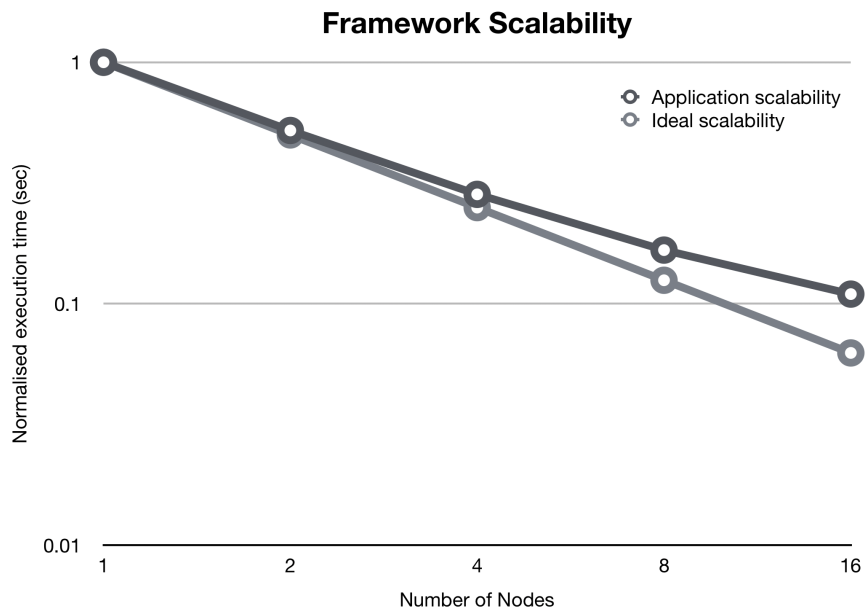

Figure 13: Framework Scalability: execution time versus number of nodes.

[12] Robin J. Wilson *Introduction to graph theory*, Fifth edition 2010

[13] John L. Hennessy, David A. Patterson, *Guide to Convolutional Neural Networks. A Practical Application to Traffic-Sign Detection and Classification*, First edition, 2017

[14] Loris Foresti, Pascal Horton *Filtres et convolution*, UNIL, 2015

[15] Hamed Habibi Aghdam ans Elnaz Jahani Heravi *Computer architecture: a quantitative approach*, Fifth edition, 2012, p. 347