ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERIA INFORMATICA

IMAGE SEGMENTATION WITH THE GROWING NEURAL GAS

Realizado por

Sanjay Prem Belani

Dirigido por

Ezequiel López Rubio

Departamento

Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

Málaga, Junio 2015

Abstract

Over many years algorithms have been proposed as methods for the segmentation of images. Everytime the work has been improved and optimized for better results with much faster algorithms and newer ways to adapt the algorithm to the needs of real world requirements. In this article we use the Growing Neural Gas on RGB images and carry out various forms of experimentations to analyze the segmentation of RGB images by this novel algorithm. There are two phases to the segmentation process where in the first phase the input data is learnt by the neural network to produce a model, and later the second phase includes the exploitation of this model to produce the segmented images. Experimentations include changing the factors that affect the algorithm and also the segmentation process.

Key words: image segmentation, Growing Neural Gas, unsupervised learning, clustering

Durante muchos años se han propuesto algoritmos que se usan como métodos para la segmentación de imágenes. Cada vez el trabajo se ha mejorado y optimizado para dar mejores resultados con algoritmos mas rápidos y con nuevas formas de adaptar los algoritmos a los requisitos del mundo real. En este trabajo usamos el Gas Neuronal Creciente sobre imágenes RGB y realizamos varios experimentos sobre ellas para analizar la segmentación de dichas imágenes con esta nueva técnica. Habrá dos fases de la segmentación, donde en la primera fase la red neuronal aprende con los datos de entrada para producir un modelo, y la segunda fase incluye la explotación de este modelo para segmentar las imágenes. Se han realizado experimentos en los que se cambian algunos de los factores que afectan al algoritmo y también al proceso de segmentación.

Palabras claves: segmentación de imágenes, Gas Neuronal Creciente, aprendizaje no supervisado, agrupamiento

Contents

1 Introduction

1.1 Motivation

Recently over the past few years we have had many technological changes in the society and a big part of that referred to the use of automated processes carried out by the modernized Turing machines (or computers as they are referred to these days). Clearly this has been possible due to the advances in the software systems and the hardware systems. Efficiency has always been an important factor considering the fact that every new upgrade brings in some more weight in terms of complexity. Therefore we try to reduce this by the optimization of codes or in rare cases creating new algorithms that are more efficient for the specific provided tasks.

Having realized that the computers seem to carry out these tasks efficiently and at a much quicker speed, we automatize the processes so we could extract as much as information that is possible to analyze the data. This way a machine would have a variety of information to determine what the image contains and how to make use of the information for further tasks. This could be used in many ways to carry out daily tasks much more efficiently and giving the possibility of the least errors as machines would simply follow an algorithm and terminate its task with reliable (in most cases) results.

Having considered this fact, we move on to the idea of our project based on image segmentation. Machines do not tend to see what the human mind perceives and interprets as images. Images for machines tend to be just a long finite series of 0's and 1's and therefore we have to give it a semantic and a syntactic interpretation so that it's able to extract and manipulate the data. At various occasions we have to determine the information an image provides us and interpret the context of the image to be used. For example, an X-ray of a patient would be used to determine (depending on where the X-ray has been carried out) where a fracture has taken place or determine the site of tumor cells that could lead to a possible destruction of brain leaving a patient paralyzed. Human eyes may sometimes may not be able to determine the tumor due to the size but if a computer would be able to determine, it could warn the medic about the consequences of the tumor that has been found.This is seen in Fig 1[10].

The above was a simple example as in to how machine/computer perception of a 2-Dimensional representation of the human brain would help save a life in most cases. We could extend this idea further to the fact that some images could be analyzed to read what an image says. For example, the writing of a human could be analyzed from a human for various purposes. Some of these may include from solving simple equations to analyze the handwriting to determine if a contract was signed by the same person or not.

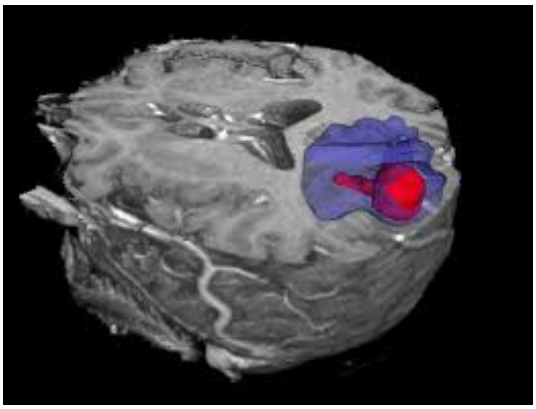

Figure 1: Shows the location of a tumor in the brain after a scan of the brain and applying the image segmentation

This is what has motivated us and allowed us to extend our idea further of segmenting an image. The algorithm of Growing Neural Gas would allow the machine to extract information on the image data that has been fed to it and use this image. By varying different parameters we could determine the optimal segmentation for an image that would be accurate and useful for a computer to interpret and analyze given the circumstances.

1.2 History

The history of image segmentation could be traced by to around half a century ago where an operator called Roberts Operator was created that was used to detect the edges between different parts of an image. This detector was the first that was able to separate the image into its constitutional parts. So this idea was the one that elaborated the definition of image segmentation and the definition that is still being used by many scientists over the past. Image segmentation could be described as a process which subdivides an image into constituent parts that extracts the parts (or objects in many cases) that are of a particular interest to the investigation.This idea has been developed over many years and in many ways where we have heaps of image segmentation taking place. It has been implemented in various machines but most have the same base that was cited in the paper of Zhang published in 1995. The process was coined as Image Engineering where it had three layer listed as below

1. Image Processing (Low Layer)

2. Image Analysis (Middle Layer)

3. Image Understanding (High Layer)

And we would wonder where does image segmentation come in place? Well image segmentation comes in effect in the initial stages of the image analysis and clearly one of the most crucial parts that would later affect the results. Image segmentation would consist of extracting the data from the material that would then be used in the object representation and feature measurement. A more visual orientation this is shown in Fig 2[9].

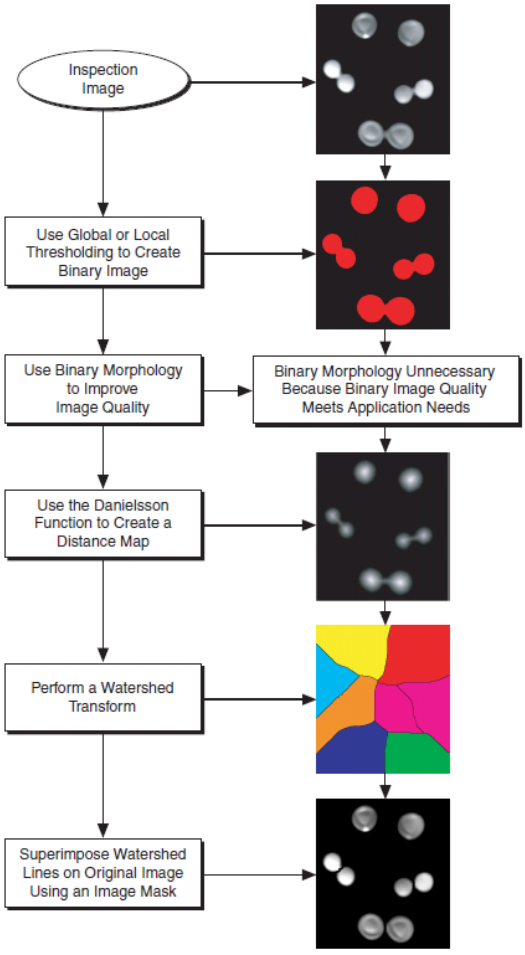

Figure 2: Shows the process of image analysis and segmentation at different levels.

The definition of image segmentation over the past has clearly changed to a point where it has been revised by various scientists, investigators, tutors and many more. Image segmentation, not being an easy topic, is still in the process of being further researched and scientists continuously provide more efficient and useful algorithms to solve the daily problems using this technique. Meanwhile the extension of 2D images to 3D images has also motivated to finding new parameters and ways to segment images and extending the idea to further dimensions allowing the process

to spread widely in the field of technology. However in spite of all these decades of investigation and research, image segmentation remains a challenging research topic.

How exactly would we describe image segmentation though in all its entirety? If we consider image segmentation as the constituent parts of the image that have been split for further investigation but the reverse process of joining these parts would give us back the original image, image segmentation would have some specific rules that are stated by Garlicky in the year 1985 as follows:

1. They should be uniform and homogeneous with respect to some characteristics;
2. Their interiors should be simple and without many small holes;
3. Adjacent regions should have significantly different values with respect to the characteristic on which they are uniform; and
4. Boundaries of each segment should be simple, not ragged, and must be spatially accurate.

But how have the definitions of image segmentation developed over years? Here are some of the definitions proposed by the great minds and developed and refurbished over the decades:

- "Segmentation is the process of partitioning an image into semantically interpretable regions." - H. Barrow and J. Tennenbaum, 1978
- "An image segmentation is the partition of an image into a set of non-overlapping regions whose union is the entire image. The purpose of segmentation is to decompose the image into parts that are meaningful with respect to a particular application." - R. Haralick and L. Shapiro, 1992
- "The neurophysiologists' and psychologists' belief that figure and ground constituted one of the fundamental problems in vision was reflected in the attempts of workers in computer vision to implement a process called segmentation . The purpose of this process is very much like the idea of separating figure from ground ..." - D. Marr, 1982
- "The partitioning problem is to delineate regions that have, to a certain degree, coherent attributes in the image. We will refer to this problem as the image partitioning problem . It is an important problem because, on the whole, objects and coherent physical processes in the scene project into regions with coherent image attributes. Thus, the image partitioning problem can be viewed as a first approximation to the scene partitioning problem ..." - Y. LeClerc, 1989

But as we there are always discrepancies with the natural language and understanding we could always apply a mathematical definition that would get rid of all the redundancy provided by the natural semantics of the language henceforth giving us a much more clearer vision at what we are trying to define. This definition was proposed by Fu in 1981 and its as follows:

- Given region R and uniformity criterion U, define predicate $P(R) = True, if \exists a \ni |U(i,j) - a| < \varepsilon, \forall (i,j) \in R$

- Partition image into subsets R_i, i = 1, ..., m , such that

- Complete: $Image = \bigcup R_i, i = 1, ..., m$

- Disjoint subsets:$R_i \bigcap R_j = \varnothing, \forall i \neq j$

- Uniform regions: $P(R_i) = True, \forall i$

- Maximal regions: $P(R_i \bigcup R_j) = False, \forall i \neq j$

1.3 Clustering and Segmentation

So now that the basic of what the image segmentation involves we could dwell deeper into the topic of clustering and segmentation. We have been defining image segmentation in various types of ways and how the definition has evolved over the years but how could we define segmentation? There are basically two types of approaches to carry out the segmentation amongst many hybrids, but the basic ones are:

1. Approach 1: Segmentation where an image is divided from the original image to smaller images recursively forming multiple components from these images. Some examples may include algorithms Divisive Clustering, Partitioning and Splitting.

2. Approach 2: Segmentation where an image is divided starting from small groups of pixels to form much larger clusters recursively forming the constituent parts that would be processed further. Examples include Agglomerative Clustering, Grouping and Merging.

The over approaches are known as a special type of segmentation called segmentation by clustering. Clustering is a process that involves formation of various groups of individuals within a specified region that have similar characteristics and properties. In most of the cases of segmentation it would include the distance between the color of the images of the standard deviation etc. To see this better we could have a look at Fig 3[8].

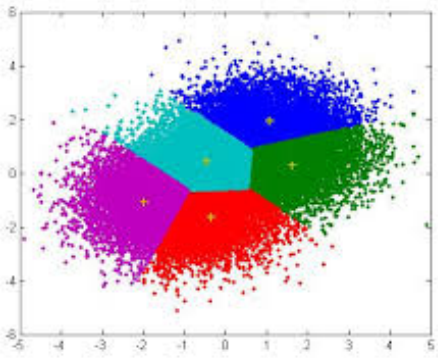

Figure 3: Shows how an algorithm would output a result after clustering different categories on a 2 Dimensional plane

1.4 Types of Algorithms

Having looked at the various kind of definitions of segmentation and having we could have more of an insight by theoretically viewing some of the algorithms that have been recently developed by many of the scientist in the past decades and use them as a basis to demonstrate the different techniques and how each of them have a different effect on the overall model. Let us look at some of the algorithms that were proposed and how they had a varying affect on the images that were to be segmented.

1. *Threshold Technique* is a technique that was widely used for a long period of time in the past due to its simplicity. This method used probabilities for segmentation where each pixel was giving a value of either 0 or 1 and this was based on the if the pixel surpassed a threshold value in the regions. This was known as gray-scaling an image or what is commonly referred to as black and white images such as shown in Fig 4[11].

Figure 4: Thresholding technique applied on the famous image of "Lenna" comparing the original image (right) and the segmented image (left)

2. *Edge-based Technique* as it has been mentioned previously was developed in the early stages of the discovery of image segmentation era too. This approach isolates images by distinguishing the outlines of each object, differentiating them from the background and hence providing us with the constituent part that is the most important to us during the segmentation process. A simple example of this could be the identification of objects in an image where just the object and its shape would matter and not the rest of the background and the redundant areas (such as in Fig 5[6]). This technique can be very effective for images with sharp contrasts, but is not as useful for blurry images and broken outlines.

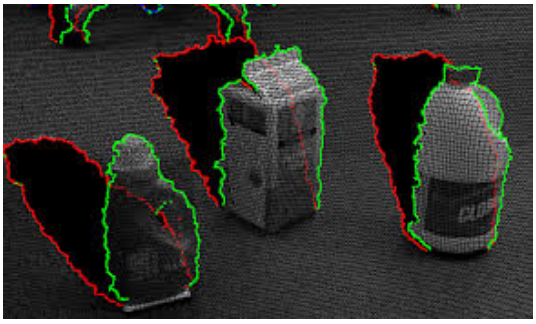

Figure 5: Showing an image segmentation based on the Edge-Based technique. This shows how the image takes the "edges" to segment the constituent parts of the image.

3. *Region-based Technique* could be considered an extension or a step forward to the Edge-based technique where not only the object taking is taken into isolation but also at the same time the isolated object is further segmented into groups and isolates the regions of the object based on the characteristics(for example in Fig 6[12]). A usual example of this is the usage of this technique in making digital art for a more precise and meticulous creation showing the creativity and the originality of the artist by exploiting the technique to its finest usage.

Figure 6: Another image segmentation of the image "Lenna" using the Region-based technique where the regions are distinctly visible.

4. *Active Contour model* is one of the most recent techniques that has been developed over the past decade. The idea is based on making an objects outline more obvious and standing out so it uses curved lines called "snakes" which "fit" around the object. This technique is useful and effective in many ways due to its versatility to be used on irregular shapes and outlines as the snakes would conform automatically to the shape of the object distinguishing our segmented region more effectively. It is also used for noisy and grainy images that affect the vibrancy and the color of the primary object. Fig 7[7] represents a simple example of this technique on segmented image of a zebra.

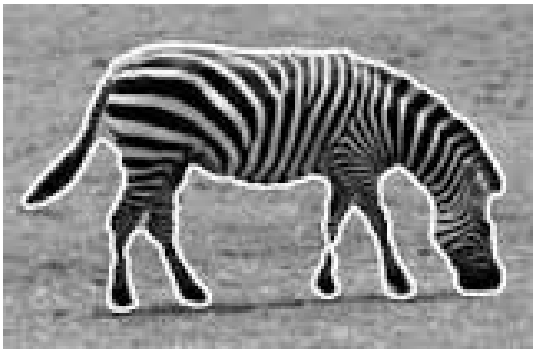

Figure 7: Image segmentation technique based on the active contours. As we can see the white contour has spread very smoothly across the outline of the zebra.

Having looked at all these different types of techniques let's have a look at the algorithm we are going to be focusing on and relate it to the per-established models.

1.5 Applications and uses of these algorithms

Having looked at all the various types of algorithms and techniques that have been extended and implemented we could look at how these have been useful in the real world applications. Some of these applications are:

Handwriting Recognition : This is one of the first applications that used the segmentation and analysis of the image to the recognize writings to compare if one writing matched another or not. Clearly the first steps involved recognition of the letters that would be later formulated and translated into a binary code that is recognizable by the computer and could be interpreted easily. Even though being two completely different applications these have developed recently over the past many years to decide the forged signatures. The detection and interpretation algorithms

have been used to create applications that read and interpret an equation and analyze and find a final solution to the given problem.

Facial Recognition : Facial recognition has turned out to be a crucial factor in the real world applications with the upgrading technology where the geometrical analysis of the face would allow systems to be locked and unlocked. Many of the advanced features also involve the elimination of the hue and saturation to a point to create a monotonous image for every individual that would recognize the facial pattern not only using the geometrical analysis but also using a color based analysis with some errors that would be taken into account when creating the base image (for example using heat detection given off by humans as show in Fig 8[2]).

Biometrics Recognition : Biometrics have become a useful weapon for government analysis in many cases where the identity of the person has to be involved and therefore fingerprints and DNA samples are taken and analyzed to give each individual a unique ID in the state. All this data could be stored in a personal identification card that would identify each individual uniquely based on the biometrics of each and therefore solving many of the problems in identifications. These biometrics could also be used as "pass-codes" as only the person containing the correct biometrics would be able to access his/her data.

Tumor detection in brains: One of the most crucial advances with image segmentation has been taking place in the world of medical science where the patients often have to be treated with easy and efficiency for the benefits of themselves and the government expenditures. Image segmentation techniques have been applied to the brain to identify tumors and the affected areas and these have been modified and expanded to know the level of harm that was caused and the areas that would have to be treated.

These are only a few examples of the advances and constantly the more efficient and useful algorithms are being used on a daily basis to improve the day to day task carried out by humans and also simplifying the process automatically at a much efficient and rapid level.

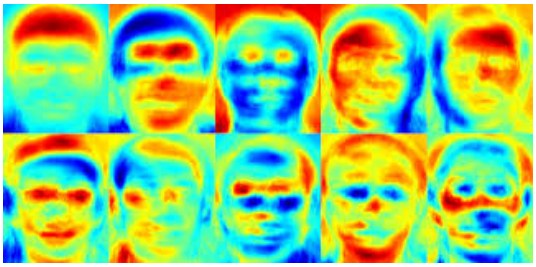

Figure 8: Face recognition based on the heat using image segmentation.

1.6 Growing Neural Gas

Now that the we have introduced the basic concepts of segmentation and clustering we could elaborate our algorithm for segmentation that has been implementation. The algorithm is based on what we define as neural networks. An artificial neural network is a connectionist computational system that connects various models together to learn information and classify or predict from a given set of data. A neural network "physically" has two main components, which are: The node and The connections. Each node is connected to another node by a connection that has a an arbitrary unit of priority designated to it which we define as the "weight" of that connection. A neural network is usually made up of many layers of these connections where the initial nodes or "sensors" receive the data from the external environment. This data is then manipulated and by the means of calculations it is transformed and learned by the nodes. Neural networks also have to tend to have layers and therefore we could categorize the important layers of the neural network into three main categories. These are:

Input Layer - Receive the sensor stimuli, data input and passes them on to the hidden layer

Hidden Layer - Receive the data and use it to learn in the several layers of neurons that may exist

Output Layer - Give the final result in the form of classification, prediction etc. using transformation functions.

These are depicted in the next image (Fig 9[13]).

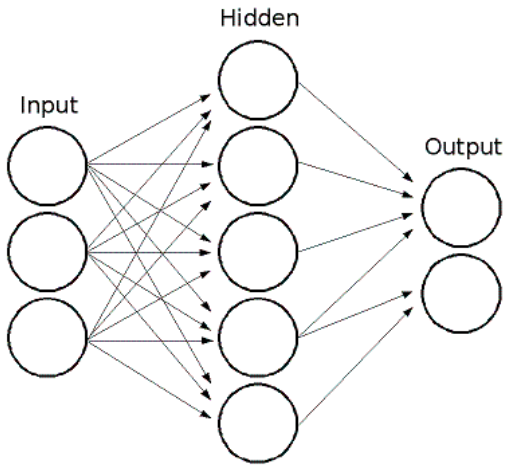

Figure 9: The basic neural network showing the three basic layers, where the second layer could be as complicated as possible.

There are various algorithms that allow a neural network to learn and develop. For simplicity we could classify the algorithms in two different types, Supervised learning algorithms and Non-Supervised algorithms. Supervised algorithms are usually "supervised" in the sense that they

have additional data and information on the output layer to allow the neurons to learn those specific categories. A simple example of this would be classifying the colors of an image where the algorithm would classify specific parts of an image into closest chosen RGB configuration given my the user. Another real world application could be the prediction of a disease at the hospital, that is, if a patient, given specific symptoms is suffering for the disease being tested or not. The output is known and therefore it is "simpler" for the algorithm to determine the classification and prediction.

The second type of algorithm that are usually taken into account are the Non-Supervised learning algorithms. These algorithms have been given the input data like the previous case however have no information on the output results and hence they have to learn and determine the output result(s) depending on the input. For instance, clustering individuals based on demographics might result in a clustering of the wealthy in one group and the poor in another. Another example is what we shall be using an unsupervised algorithm for and that is image segmentation. Given an image with the color configurations of RGB, and depending on some parameters that shall further on be explained in the article, how many "clusters" or "segments" can be deduced simplifying the image breaking it into its more constituent parts to be analyzed later. Here the number of output segments is unknown and therefore we would find out the prototypes of each region (The pixel that is the configuratively based the closest to the remaining pixels in the region) and the number of regions that have been formed using these parameters.

Growing Neural Gas is based on the idea of a Self-Organizing Map (aka SOM) which is a learning method that uses vector quantization for the visualization of data and as well as the clustering for a more accurate and global vision on the data that has been presented. As stated before, these maps would be topologically ordered and would allow the formation of clusters of neurons. This idea of this algorithm was first introduced in early 1990s where it was not "growing" but there were some parameters passed to the neural gas network an the algorithm learnt based on those parameters. In this case there are no parameters passed to the algorithm and therefore the leaning process continues over time until a performance criterion has been met. The algorithm as described above has various uses such as quantization, clustering and interpolation.

Now that we have introduced what the idea of Growing Neural Gas, we could look at the technicalities to comprehend the better functionality of the algorithm. The Growing Neural Gas Algorithm proposed by Bernd Frtizke[5] is an unsupervised algorithm that is incremental with time and cycles depending on some of the key parameters that shall be discussed further. algorithm. Given some input distribution in R^n, GNG incrementally creates a graph, or network of nodes, where each node in the graph has a position in R^n. GNG algorithm can be used to find the topological structure that would closely reflect the inner structure of the initial input data. Also one of the many advantages of the GNG algorithm is its adaptability. Over time GNG can adapt to new additional data or modified or changed data over time covering the new distribution with more nodes.

The algorithm initialized with two inputs, node n1, and node n2 that are initially connected

together. The algorithm develops over time where two nodes are neighbors if they are connected by an edge. The determination of which nodes are classified as neighbors or not is dependent on a variant of the Hebbian learning CHL that is,

"For each input signal x an edge is inserted between the two closest nodes, measured in Euclidian distance"

This is an essential part of the algorithm because its calculation will lead to the adaptation of the nodes and the addition and deletion of nodes. Of course now we could perhaps ponder on how would we determine the number of nodes in the resulting graph to be produced. In this case as the graph is dynamically generated the number of nodes is incremented during the execution of the algorithm. However for efficiency purpose it is always a good idea to add a maximum number of nodes to be created because the algorithm could execute infinitely to reach the minimum error that would halt the execution of itself. Another solution could be another performance parameter determined by the user initially, which once reached, would halt the addition of new nodes. The main pseudo algorithm for the GNG is as follows (Fritzke):

1. Two nodes with the vectors of weights are created, allowed by the distribution of input vectors, and zero values of local errors; connect nodes by setting its age to 0.

2. Then generate an input signal *x* conforming to some distribution

3. Find two neurons *s* and *t* nearest to the input signal *x* previously generated, ie. nodes with weight vector w_s and w_t such that is $||w_s - x||^2$ minimal, and $||w_t - x||^2$ is second minimal value of distance among all nodes.

4. The winner-node *s* must update it's local error variable so we add the squared distance between w_s and *x* , to our errors variable E_s :

$$E_s \leftarrow E_s - ||w_s - x||^2$$

5. Shift the winner neuron *s* and all of its topological neighbors (ie, all neurons that have a connection to the winner) in the direction of the input vector *x* by distances equal to the shares e_w and e_n from a full one

$$w_s \leftarrow w_s + e_w \cdot (w_s - x)$$

$$w_n \leftarrow w_n + e_n \cdot (w_n - x)$$

6. Increment the age of all edges from node *s* to its topological neighbors by 1

7. If the two best neuron *s* and *t* are connected, set the age of their connection to zero. Otherwise create a connection between them.

8. Remove connections with an age larger than age_{max} If this results in neurons having no more emanating edges, remove those neurons as well.

9. If the number of the current iteration is a multiple of λ, and the limit size of the network hasn't reached, insert a new neuron *r* as follows:

   (a) Determine a neuron *u* with the largest local error

   (b) Determine amongst those neighbors of *u* the neuron *v* with the maximum error

   (c) Insert the node *r* between the nodes *u* and *v* with the following criterion:

$$w_r = \frac{w_u + w_v}{2}$$

   (d) Create edges between *u* and *r*, and *v* and *r*, and then remove the edge between *u* and *v*.

   (e) Decrease the error-variables of *u* and *v* and set the error of node *r:*

$$E_u \leftarrow \alpha \cdot E_u$$

$$E_v \leftarrow \alpha \cdot E_v$$

$$E_r \leftarrow E_u$$

10. Decrease errors of all the neurons *j* by the fraction β.

$$E_j \leftarrow E_j - \beta \cdot E_j$$

11. If the stopping criterion is not met then repeat from step 2.

1.7 Explanation of the parameters

During the algorithm we there have been many parameters that we change for the execution of the algorithm. Some of the parameters may seem obvious whilst the others do not and therefore here are some of the crucial factors of the algorithm that could change the overall output. Some of these factors have been experimented with to establish some general basis and how the change affects the results.

"Errors" - During the execution one of the most crucial aspects to keep the algorithm running is the usage of errors. During the algorithm we try to maximize the error because we want the

clusters to be as separate as possible and therefore in the beginning of the algorithm we update the error as the squared distance to the input. This gives us a way of detecting nodes that cover a larger portion of the input distribution. Later on in the algorithm the value of all the errors are decreased to give more importance to the most recent calculated error.

"e_w" and "e_n" - These are factors that affect the movement of the winner node *s* during the execution of the algorithm. These parameters affect the overall placement of the winner node and therefore are not really established with high values as that would lead to an unstable graph where at every execution the nodes would be making big movements. This could change the overall aspect of the graph and never reach an equilibrium. The values that have been experimented and tested with suggest the value of e_n be a hundredth of the value of e_w as these have performed well in most of the given conditions.

"age_{max}" - The priority and the validity of the edges have to be modified at some point in the graph by the algorithm and the variable age_{max} is responsible for those purposes. The local aging process can be used to invalidate the ages as during the execution of the algorithm the affected edges are set with an age factor to zero and the latter, when reached a factor called age_{max} are reached, are invalidated from the graph. At some point there could be a node that would/could be left without any edges connecting to it. These nodes (referred to as "dead nodes") could be removed from the graph as they would not affect the structure of the graph at anytime during the execution of the algorithm.

"λ" - Edges could be inserted or removed however during the execution of the algorithm and due to its adaptive nature a new node has to be added at some point and the parameter λ determines this. Every λ^{th} iteration a new node is inserted between the node with the largest error and its neighbor with the largest error. It might not be the wisest policy to have a fixed point at which the node is added because it might be unnecessary so a desirable policy could be set (for example depending on the local errors). Initial settings in a determinate case would matter as setting it too low would result in most of the nodes being added in the earlier stages of the algorithm resulting in a poor distribution of the nodes. Another affect a low factor could have is the generation of "inactive nodes" which would not reposition themselves around the graph too much as their values would have been restricted due to the initial stages of the algorithm and the newer inputs would not approximate the already established node values.

2 Methods and tools

The entire implementation of the algorithm has been carried out with a widely known software called MATLAB which has its own language and implementation to be based on and carried about. MATLAB in itself is a high performance language that is widely used for the technical computing. One of the main advantages of the language is that the mathematical notation could almost literally be programmed in the same manner as it is written. This is a big advantage over the usual programming language where there has to be a process of "unrolling" the equation to allow the language to analyze is syntactically. Some of the uses that MATLAB facilitates are :

- Modeling, simulation and prototyping
- Scientific and engineering graphics
- Algorithm development
- Data analysis, exploration, and visualization
- Math and Computation
- Application development, including Graphical User Interface building

The language allows *both programming in the small* to rapidly create quick and dirty throw-away programs, and *programming in the large* to create complete large and complex application programs. The programming language library also contains a vast amount of functions and computational algorithms that facilitate the user whilst making quick and functional programs.

Some of the main functions that we have used as an aid to carry out most of the tasks in our implementation our :

- imread(*filename*) - This function takes an image and transforms it to an adequate data structure understandable by the programming language. It infers the format of the file by its content and on general basis it accepts most of the known file formats such as *.tiff, .jpeg, .jpg, .png* etc.
- reshape(*A,[m n]*) - One of the most important aspects of the programming language is the usage of functions that allows the re-dimension of a matrix without affecting its inner contents drastically keeping the same order. The GNG algorithm used would be taking in very specifically structured matrices and therefore the re-dimension of the input image has to match the algorithm for coherent results. The parameters include the matrix and the array contains the size of the matrix that it would be transformed/redimensioned into. Matrix A would be redimensioned into a matrix of size m-by-n.
- nnz(*A*) - Sometimes a matrix may contain a number of zero elements and our interest may rely on only the non-zero elements. This function takes in a matrix and returns the number of non-zero elements in the matrix.

- spfun(*fun,S*) - This function takes all the nonzero elements of S and applies the function "fun" to them. This is useful when when applying a repetitive functions for example in an array or in a matrix. In our case we would be using a descending function to be applied on the Vector S that would allow an ordered list in the descending order.

- find(X) - Finds the all non zero elements of the array X which could be important as it would allow us to distinguish the main elements without having to evaluate the vast majority of zero values that do not concern us

- graphconncomp(A) -¿ This function would allow us to find the number of connected graphs in the image hence showing us the number of segments formed by the GNG algorithm from the input image that had been provided

- imshow*(A)*/imagesc(A) - Takes A and displays it as an image. It could be classified as the reverse function for imread

The experiments have been carried out using a Windows 7 Home premium Operating system with a 4GB RAM and an Intel i7 2º Generation CPU. All kinds of graphics have also been designed using the MATLAB programming language.

3 Implementation

The implementation of the code has been carried out using MATLAB as discussed above and therefore there are various files in the project that we shall now discuss. There have been two main approaches that were followed during the project when the implementation was carried out.

- The first implementation for would include the color combination RGB where the segmentation would only be carried out using the RGB values and the prototypes would be evaluated based on the winners that are closest to the neuron with the most synaptical weight (winner neuron for the area).

- The second implementation could be considered as an extension to the first implementation where not only the color combination of RGB will be taken into account but also the position of the pixel would be an important factor (even though it will still be at a lower priority than the original RGB weights). This would prevent the overlapping of certain areas in the images which will be explained in the Results section.

3.1 Training algorithm for the neurons (Growing Neural Gas)

This file is used for the implementation of the growing neural gas algorithm. Samples of data is passed to the algorithm with the *n* dimensions and the function generates a corresponding output with some predetermined parameters that have been passed to the algorithm. The function is called from one of the *DemoGNG.m* files depending on how many dimensions would be taken into account when carrying out the learning process using the grown neural network. The algorithm is affected by many of the parameters here we briefly explain how each of the parameters has an effect on the algorithm implemented:

- *Samples* - These are the samples that are going to be passed to the algorithm. The original image for the algorithm is read using the *imread* function whose functionality has been previously explained. This function returns the image as integers which we then use to convert it to doubles and scale it to our necessity. The image returned by the imread function is in three dimensions where the size of the three dimensional matrix would depend on the size of the image in its width and height and the third dimension depending on its type of image (for gray-scale images it would be 2 dimensions and for an RGB image it would be 3 dimensions).

- *MaxUnits* - This represents the number of maximum nodes that would be added to the resulting graph once the algorithm is in execution. As stated previously the factor could be important depending on the complexity of the image. It also helps provide efficiency and optimization for the execution as having numerous nodes may not be the best approach as there maybe little change when executing the algorithm.

- *Lambda* - Is one of of the key factors and the simplest parameters in the GNG algorithm which could be extrapolated and made as complex as possible for the execution of the algorithm. This factor controls how often a node in the graph would be added to the algorithm. We pass it as a parameter due to several reasons. One of the first reasons is for the simplicity where the user can decide into how often the algorithm would allow the addition of the new node. However we could also make the addition of the newer nodes dynamic rather than static in the near future as the additions of nodes, as stated previously, statically might not be the best case scenario for optimum results.

- *EpsilonB* - This parameter is used for the translation of the winner neuron to be moved in the specific position when updating it after the arrival of the newest input. This factor is usually between 0 and 1 and is used as a scaling vector when translating the winner neuron. The movements of the winner neuron usually tend to be linear and therefore the greater

- *EpsilonN* - This is one of the factors that affects the the neighboring neurons to the winner neuron in a similar way. This is used as a scaling factor for the translation of the neighboring neurons however it is set to a much lower value than the original scaling factor of the winner neuron resulting in a smoother behavior. Due to the similarity in the nature, this factor also tends to be in the value between 0 and 1.

- *Alpha* - This factor affects the subsequent decrease in the error variables of the nodes u and v when a new node is added considering that the error values for these nodes should be invalidated. This is important taking into account that the errors would be decreased as the the we would not want the next node in line to be added in the same area or zone and therefore we decrease or invalidate the value of these nodes. Also has we have previously stated, the error values represent the coverage of the nodes and therefore as the new node already has assumed some coverage of the input distribution we invalidate the previously defined nodes.

- *AMax* - AMax represents to the maximum age factor of the edges and here we pass it as a parameter to remove an edge when it crosses this value as for the reasons stated above previously.

- *D* - This is theβ factor that is used at the end of the algorithm which is the decrease in the global errors. This is carried out in order to give priority and importance to the most recent errors that were calculated and also to control the local errors growing out of proportion.

- *NumSteps* - Due to the iterative and infinite nature of the algorithm we cannot determine the point where it would stop and therefore we always need a terminal point for the algorithm to terminate its execution. This factor determines the number of iteration of the algorithm before it trains the neural network following the pre-established parameters. There are other ways to determine the termination of the algorithm, for instance, defining a minimum error

to after which an algorithm stops or a few executions after the last node is added to the algorithm. These are always depending on the extent of complexity the algorithm would be implemented to. We could juxtapose various types of terminating factors to outline the most optimal ones in the near future to determine the best outcomes.

The algorithm that is implemented in MATLAB follows the pseudo-code in a coherent manner, saving the most relevant information in a data structure that called *Model* which has various attributes, mainly the ones corresponding to the parameters that have been passed to the function previously. Besides the parameters, we add three more attributes to the model which are *Means, Errors* and *Connections. Means* represents all the prototype vectors that or also known as the "winner" neurons for our implementation. The *Errors* would represent the global errors that are taken into account throughout the algorithm. Finally the *Connections* attribute represents the edges of connections between the neurons. This has been implemented using a matrix where we take into account the age of the connection. If two neurons are connected, the value of the cell representing the connection would be a positive integer that would represent the number of steps remaining till the connection is removed, that is, representing the time to live of the matrix. On the contrary no connection is represented by a zero value.

3.2 Local representatives

After the algorithm trains and creates the model, we need to find out the criteria to base upon to carry out the segmentation. To carry this process out for every pixel on the image we need to find the winners of those regions and also calculate the error to the winner node. To carry out this process we create a function called *CompetitionGNG.m* which gives us a matrix where the row index of the matrix would represent the pixel being represented, the first column representing the winner neuron for the region sample that has been selected. To carry out this calculation of errors we use the mean squared distances between the samples and the vector prototypes attained from the "Model" data structure which has been derived from the training algorithm previously described. We take the neuron with the least error and add it to our output matrix in the second column, hence giving us a *N x 2* matrix where N represents the total number of samples obtained from the image.

3.3 Plotting

We carry out using the plot functions that have been previously in the MATLAB language. We implement this in the file called *PlotGNG3.m*. For the plotting of pixels we use the *plot3(...)* function provided to enter the three dimensions of RGB for the neuron representing the prototype of the region. Once these have been graphically laid out, we need add in the edges or connections between these prototypes that remain from the growing neural gas algorithm. This is carried out

using the *line(...)* function where we iterate through all the nodes in the "Model" data structure and find its existing neighbors. This gives us a a three dimensional model of the prototypes of the nodes representing the mode of each RGB combinations present in the image that can be seen using the MATLAB software.

3.4 Simulation and Execution

Once we have attained the means of getting all the variables ready for the prototypes we need to carry out the segmentation of the images. We implement the functions and scripting in the files *DemoGNG3.m* and *DemoGNG5.m*. The suffixes 3 and 5 in the names of the files refer to the dimensions that would be used for the learning of the neural networks using the GNG algorithm. There are two approaches to carry out the segmentation where the first approach takes only the RGB factors into account whereas the second algorithm uses the pixel coordinates to determine the overlapping regions of the segments in the image. The reasons for this will be much more visual in the "Results" section where the algorithm would be implemented and executed on various images.

Both approaches follow a similar implementation so we start off with describing the first approach to segmentation which is the pure RGB based segmentation without taking into account the position of the pixel. Firstly we read the image using the *imread(...)* function which has been previously explained. Now due to the function returning a result in an $M \times N \times 3$ matrix, and our implementation of the Growing neural gas algorithm takes samples in the dimensions $3 \times (M \cdot N)$ matrix, we use the reshape function to transform the matrix to the desired dimensions without physically affecting its contents or the order. Next we pass it to the algorithm to be trained which returns a model as a result containing all the attributes relevant to the neural network as we have mentioned above.

Now for optimization and general purposes we remove some of the edges that have had the least time to live, or nearly had been removed or would be removed in a few iterations to have a more stable and weighted graph that has relevant connected nodes. Using the *sp-fun(...)* function we sort the connections in a descending order and then take take only the first *nnz(Model.Connections)-NumConnectionsToDelete* number of connections where we apply the *nnz(...)* function to take into account only the edges that are existent (non zero value edges). using the new updated *Connections* attribute we use the *graphconncomp(...)* function to find out the number of connected graphs in the output model to determine the number of distinct segments that have been created.

The next step is to calculate the winner nodes for each sample and the error that related the sample nodes and the winner prototype. Therefore we use the Local representatives to determine the winner nodes that has been previously implemented. Once we have the output result from the matrix we follow on to replace all the samples by their local representatives(which would result in segmenting the image as only the prototype RGB combinations would be the ones that would be

showed in the image). We reshape the image as it has been previously modified for the coherency with the training algorithm, and finally output the image using the *imshow(...)* function.

To continue experimenting different segmentations we apply the connected components factor to the winner nodes in the algorithm as well to see the distinct regions that were formed. These show the connected component samples. Due to the scattering of the image and the regions we apply some filters to elevate the smoothness the regions using the function *colfilt*(...) which basically takes a region (with a size specified in the parameters of the function) and uses a statistical function(mode in our case) to calculate an RGB combination for the pixel that is selected for that region. We modify this effects of segmentation more by using solely the prototypes rather than the samples and therefore we apply the the same kind of functions as in the previous pair of segmentation to obtained the connected components prototypes and the filtration of the connected component prototypes based on the mode of the pixel in each reason.

We used the connected component prototypes to calculate the mean squared error to result in an output and later show some results varying a few parameters (that would be more visual in the results section) to determine some optimum attributes for the segmentation.

The idea of the second approach is very similar in fact to the idea of the first approach where we carry out the segmentation purely based upon the RGB combination. In this case we would like a distinction between a color combination based upon the location of the pixel as well as its RGB combination. A simple example of this would be when a region separates two regions of the same color. In some cases, even though the divided regions have the same color we would prefer it to be classified distinctly as they might have been unnecessary segmentation. To carry out the process we use the a function to provide us an two arrays of $M \cdot N$ elements which would be the cartesian product of the dimensions of the image and would define position of each pixel in an image. These are then added to the segments sample and reshaped to be passed to the training algorithm to achieve results. Clearly the position vectors are given a lower priority to the RGB values as the priority would be determine how much the position really affects the segmentation (corresponding parameter in the file known as *SpatialFactor*).

However the plotting process in this case for the model has a change as it is not easy to plot a model with five dimensions on a two dimensional computer screen. Therefore as each prototype has a given RGB combination, we plot the model using this technique where the nodes are colored with their cartesian product and the positioned using the position vectors that were previously calculated. In this way we could represent the model in a much more visual way in two dimensions.

4 Results

4.1 Defining the parameters

Here we carry out the same techniques as in the previous approach with trying to find some optimized parameters that give well established results in most of the conditions, however using the second approach that uses the coordinates of the pixels alongside the RGB combination to segment images. Due to the various numbers of parameters that affect the algorithm and the segmentation we decided to choose one parameter from each of the phases of the project to carry out our observation to analyze some results. The other parameters were merely tested with trial and error based experimentations until some coherent results were obtained. To carry out this investigation we use the following image as a base to establish some default values.

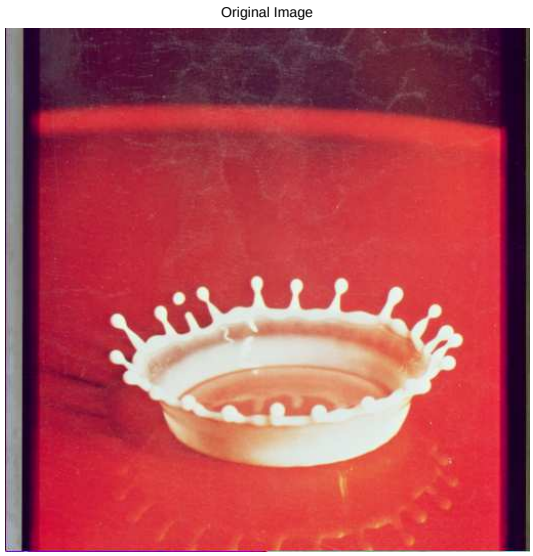

Figure 10: This is the famous "Crown" image used as a basis in many experiments using image segmentation

To carry out the experiments for our investigation we use *NumConnectionsToDelete* and *SpatialFactor* as the parameters that will be varied and tested whilst the other parameters would be kept constant to see how the image segmentation and the training is effected by the changes in these algorithms. The parameter we take into account for the investigation purposes is the *NumConnectionsToDelete* which the number of connections with the least time to live that would be deleted after the model has been established. We shall use vary this along and plot it against the mean squared error of the original pixels and the winning component prototypes that had been

previously calculated. We use the simulation file as an apparatus to provide us this result and hence leading us to create a new file called *ParametersTesting.m* that will contain the iterations needed to test the results and plot them in a two dimensional plane. The second and the final parameter is defined by the smoothening of the image that would be produced. During times of segmentation there are many regions where the segemented regions are so superficial that the segemented image might appear "grainy" or "raw". To avoid this we use the *SpatialFactor* which is the weight that we would be assigning to the pixels and therefore the algorithm would not only train the neural model based on their RGB combination but also on the position of the pixel with a certain weight. We carry out this priority of RGB combinations over the coordinates of the pixel so that the weight doesn't completely influence the segmentation that would be carried out, rather gives a guidance to the algorithm for the differing regions in the image so it could be trained more adequately giving coherent results.

We proceed with our experiment where we change the segmentation parameters rather than affecting the output of the results and hence we use specific parameters (stated in the figure reference) whilst changing only the *NumConnectionsToDelete* factor.

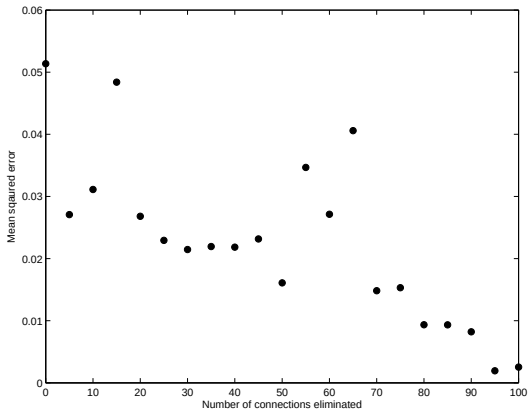

Figure 11: Figure shows the effect the change in the number of edges deleted from the model has to the minimum errors between the winning component prototypes and the samples evaluated. As it can be observed there is a gradual decrease showing a negative correlation between the higher the number of edges deleted, the lower the mean squared error calculated would result in. The parameters that were used for the training purposes are *MaxUnits = 50, SpatialFactor = 0.3,* λ *= 100,* e_w = 0.2, $e_n = 0.006$, α=0.5, age_{max}= 50 and β = 0.95.

A gradual negative correlation is noted in the graph with the mean squared error increasing in every scenario. The parameters that affect the change ranges from 0 to a 100 with increments of 5 in every calculation of the mean squared iteration. As it is visible there are some values which do not follow the correlations in the exact manner limiting us with some anomalies that are to be considered for later analysis.

To conclude our final set of parameters testing we carry out our last benchmark parameter using the spatial factor parameter. The following figure shows the result.

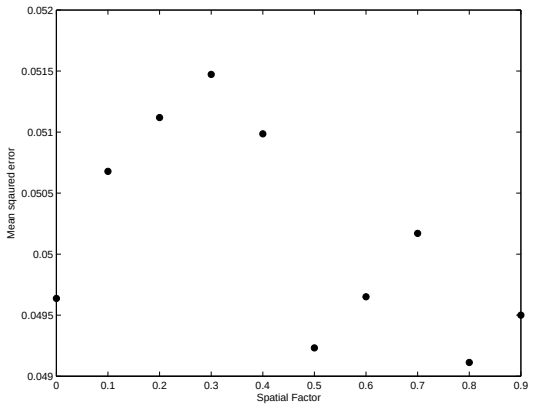

Figure 12: Figure shows the effect the change in the number of maximum nodes added to the model has to the minimum errors between the winning component prototypes and the samples evaluated. The plot shows a great decrease in the number of errors with the increase in the number of nodes. The parameters that were used for the training purposes are *NumConnectionsToDelete = 0, MaxUnits = 50,* λ *= 100,* $e_w = 0.2$, $e_n = 0.006$, α=0.5, age_{max}= 50 and $\beta = 0.95$.

We can see there is a gradual decrease in the squared error with some anomalies that seem to classify the regions incorrectly. These anomalies are due to the randomness of the algorithm in the process of addition of nodes. The steps for the change in the spatial factors are from a 0 to 1 with increments of 0.1.

This correlation is the result of the deletion of some of the nodes that were unwanted in the graph at that moment. The factor deletes a certain number of edges from the graph and therefore whilst segmenting images we could relate and deduce the fact that a deleting more edges than we actually need increases the the mean squared errors between the samples and the prototypes as the deletion of the important edges leads to incoherent results. The segmentation would be ineffectively carried out as the samples would not be adequately connected to their belonging prototypes leading to the inconsistencies in the results.

To make sure we have been guided correctly with the chosen parameters for the training of the algorithm that allows the segmentation of the image, we try the algorithm on a different kind of image. This image is another crucial image used in image segmentation techniques for the testing and analysis of algorithms whilst measuring its performance. The following is the image that will be used for the segmentation.

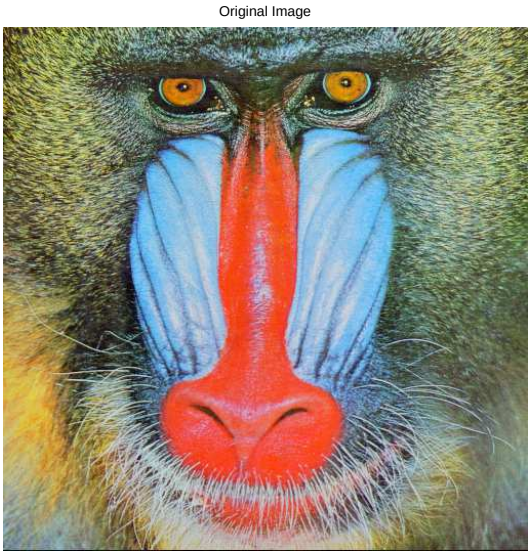

Figure 13: This is the famous "Baboon" image will be used for the further verification of the results that were obtained from the previous image segmentation performance measure.

The image has sections of brightness and shades of similar colors which is similar to the previous image hence allowing us to maintain a coherency. We carry out the same tests as the previous image. The following two figures show the changes in the parameter *NumConnectionsToDelete(Fig 14)* and *SpatialFactor(Fig 15)*.

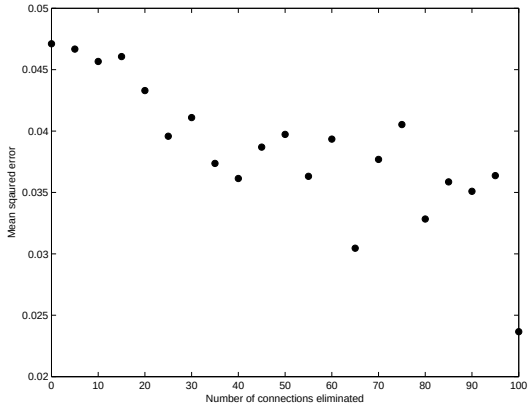

Figure 14: Figure shows the effect the change in the number of edges deleted from the model has to the minimum errors between the winning component prototypes and the samples evaluated. As it can be observed there is a gradual decrease showing a negative correlation between the higher the number of edges deleted, the lower the mean squared error calculated would result in. The parameters that were used for the training purposes are *MaxUnits* $= 50$, *SpatialFactor* $= 0.3$, $\lambda = 100$, $e_w = 0.2$, $e_n = 0.006$, α=0.5, age_{max}= 50 and $\beta = 0.95$.

And now the change in *SpatialFactor* parameter.

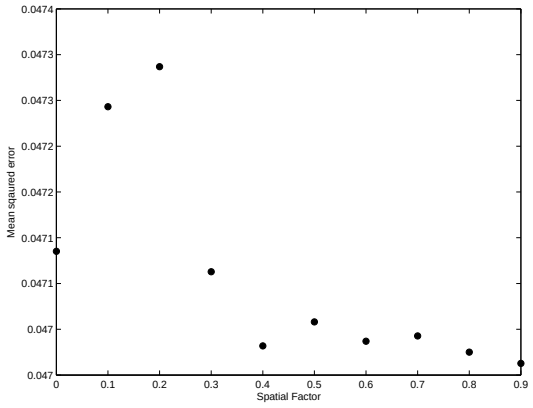

Figure 15: Figure shows the effect the change in the number of maximum nodes added to the model has to the minimum errors between the winning component prototypes and the samples evaluated. The plot shows a great decrease in the number of errors with the increase in the number of nodes. The parameters that were used for the training purposes are *NumConnectionsToDelete* $=$ *0, MaxUnits* $=$ *50,* $\lambda =$ *100,* $e_w = 0.2$, $e_n = 0.006$, α=0.5, age_{max}= 50 and $\beta = 0.95$.

As we can see the results match and the same tendency has been generated by both the segmentations that are being carried out strengthening the hypothesis that has been previously deduced based on the correlation of the graph.

4.2 Parameter testing with various images

We can determine a reasonable value of *NumConnectionsToDelete* $=$ 35 as this removes a reasonable amount of unwanted connections at the same time as reducing the error to its finest. However we cannot be perfectly sure if the applied result might be an anomaly or a result so it would be imprecise to analyze our observation only on one result. But for the generalization purposes and to carry out the investigation more adequately, we use these parameters to see the effect of segmentation on different types of images.. For the variable of the *SpatialFactor* which would be the weight given to the position of the pixels at the moment of the training algorithm we could see that a reasonable parameter would be *SpatialFactor=0.5* as it gives enough weight to the positions to have a reasonable effect on the training to reduce the result.

The experimentations would be carried out in various parts where we first carry out the simple segmentation of the image without the need of generalizing the regions. In this mode of segmentation we have the nodes defining the regions, i.e the each node represents its own region and therefore the resulting segmented image is very similar to the original input image. Next we carry out a segmentation using the spatial factor using the pseudo coloring technique. To the

naked eye the difference between many shades of the same color may not be as clearly visible and therefore the pseudo-coloring allows the the different regions to be enhanced out more clearly for the difference. This allows us to have a better grasp on the segmented regions which could be used for further analysis. Once the pseudo coloring is applied we could use the real colors instead of the pseudo-colors and therefore we carry out the segmentation and the smoothening exactly in the same way however using the prototype colors that had been previously calculated using the training algorithm. This way we have a clear comparison using the two segmented images with pseudo-colors and another two segmented images with the real colors of the image.

4.2.1 Crown Image

Here we carry out the testing with one of the famous crown images. These tests will be used on further with other images too. Here we present again the original image for comparison with the segmented images and following that would be the model generated by the training algorithm.

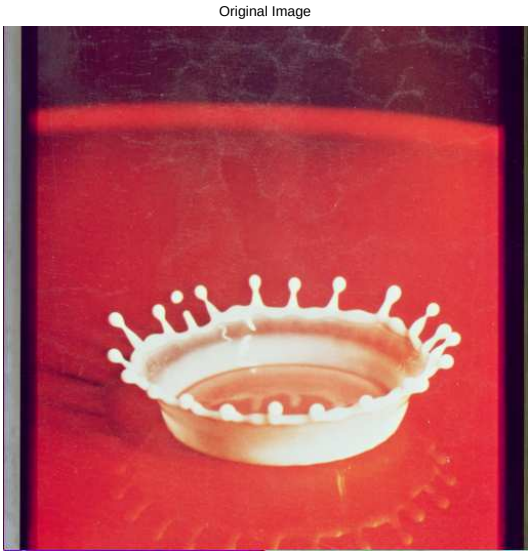

Figure 16: "Crown" image used as a basis in many experiments using image segmentation

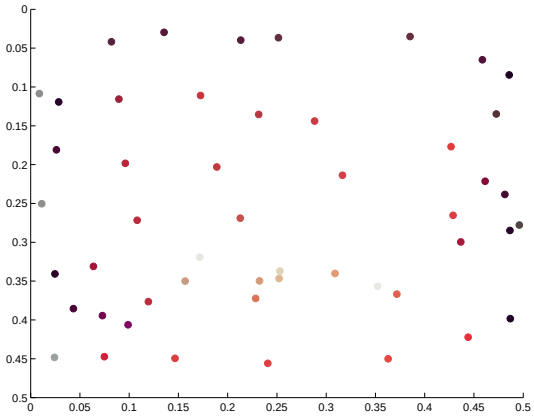

Figure 17: This figure shows the model generated in 5 dimensions however represented on a two dimensional scale where we can see the nodes are the winning prototypes of the image that was segmented and the RGB combination they are colored in shows the prototype colors. The positions represent a location of the winning prototypes.

The next figure represents the first image that is produce when we simple segment the image based on the samples and the model.

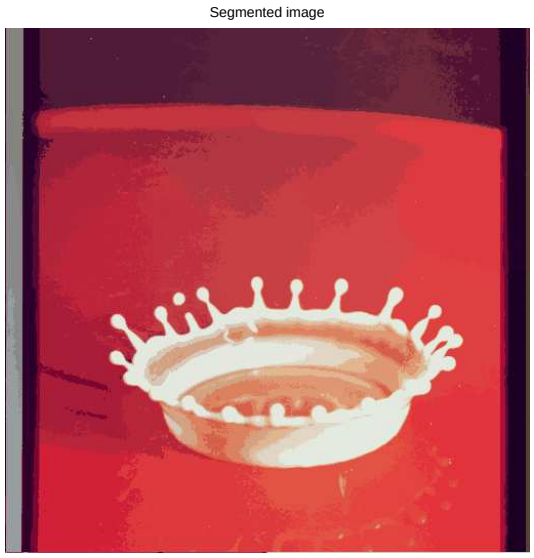

Figure 18: Figure shows the effect of the algorithm had on the segmented samples. As we can see from the original image and segmented image there are only minor differences between them.

Having looked at the segmented image we proceed to segment it further so the segmentation patterns are more visible and therefore we create an image that include the connected components

only replacing all the samples by their winner prototypes. These are shown in the pseudo colors that were previously explained in the beginning to enhance the distinction between the regions.

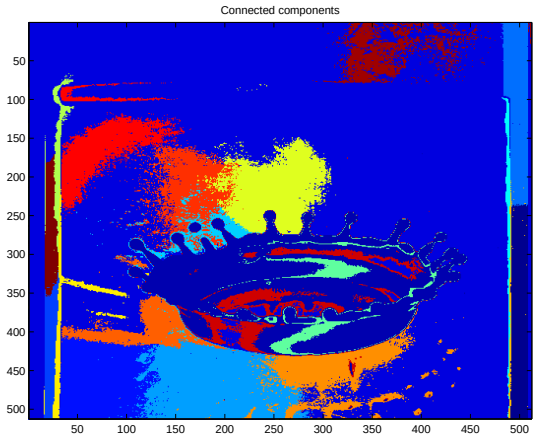

Figure 19: This image shows the connected components of the image where the samples have been replaced by their winner prototypes. This applies the pseudo-colors for a better enhancement of the regions

Clearly we can see that there is some roughness in the image as the patterns that have been segmented are raw and haven't been tampered with. Therefore to reduce this "rawness" from the image we apply the filter that we had previous discussed.

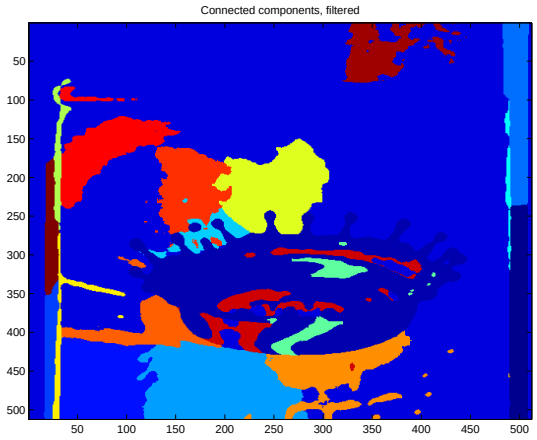

Figure 20: Figure shows the connected component images as before but having replaced the filters using a sliding window technique. In this case the parameters include a region of 7 x 7 pixels that slides across the image replacing the pixels with the most modal pixel in that window.

Now this sampling could be more refined by using the connected components prototypes rather than the samples solely. We apply the same technique as above in the following two images with the prototypes rather than the samples that were replaced. This allows the real colors of the image to be shown rather than the pseudo-colors allowing us to achieve what was our original prospective.

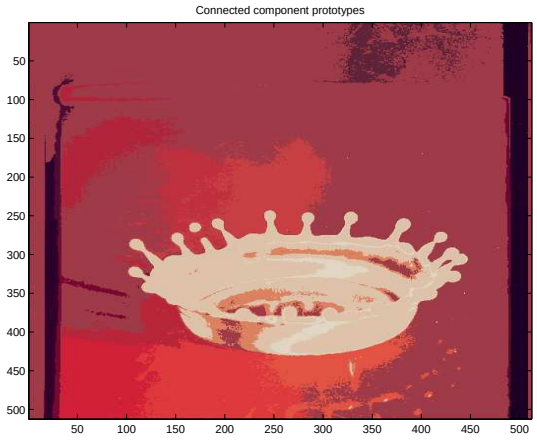

Figure 21: Connected components prototypes solely using the image segmentation techniques. The rawness of the image is pretty visible due to the "grainy" effect.

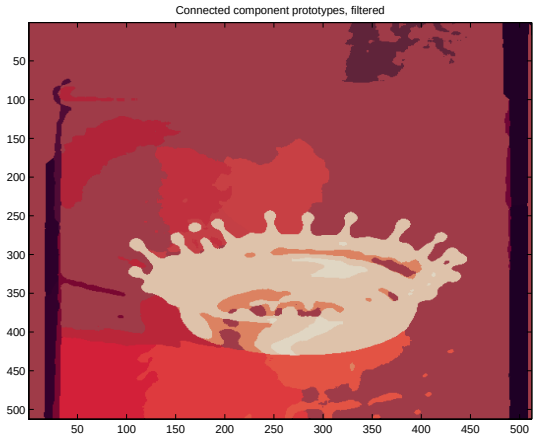

Figure 22: Here we apply the filtering to apply smoothening of the image with the replacement of the "rawness" of the data.

As we have seen from the previous images, the image segmentation has been achieved by taking many steps where we analyze the output of the image and test various parameters

4.2.2 The frog

This is another image that is widely used for the detection of edge lining of the image as it has a blurred background and a frog[4] in the foreground.

The next image represents the original image of the frog that will be segmented.

Figure 23: Figure shows the connected component images as before but having replaced the filters using a sliding window technique. In this case the parameters include a region of 7 x 7 pixels that slides across the image replacing the pixels with the most modal pixel in that window.

Now we present the model that was generated by the algorithm when the growing neural gas was applied to it.

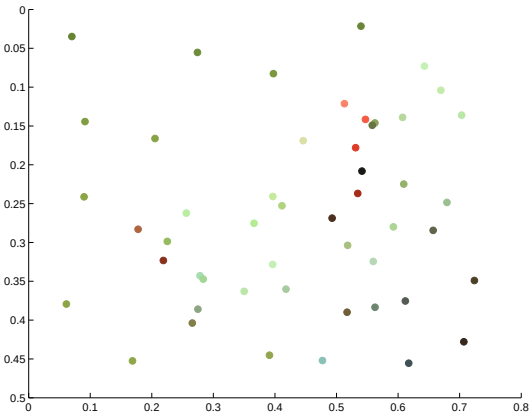

Figure 24: Figure shows the connected component images as before but having replaced the filters using a sliding window technique. In this case the parameters include a region of 7 x 7 pixels that slides across the image replacing the pixels with the most modal pixel in that window.

As it is seen in the photo, the variety of colors in the image is limited however the few colors that are present appear in different kind of shades which would help us to deduce if the algorithm works equally well in different shades of the same color or not. The next images show the segmentation of the image using the various techniques that have been seen above.

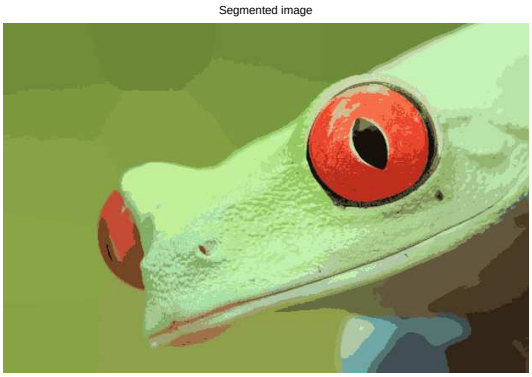

Figure 25: Figure shows the connected component images as before but having replaced the filters using a sliding window technique. In this case the parameters include a region of 7 x 7 pixels that slides across the image replacing the pixels with the most modal pixel in that window.

To create a clearer view of the segmented images we have a look at the connected samples and then we apply the filter to see the difference between the two images.

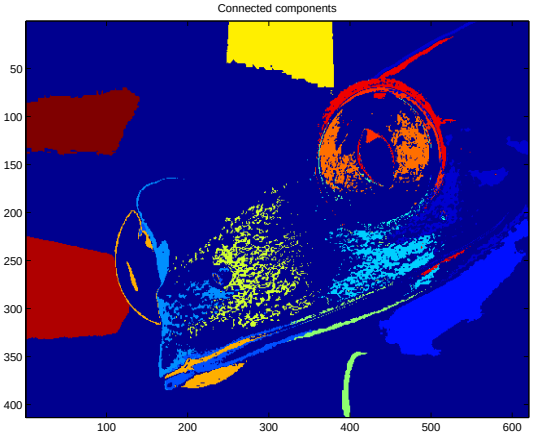

Figure 26: This image shows the connected components of the image where the samples have been replaced by their winner prototypes. This applies the pseudo-colors for a better enhancement of the regions

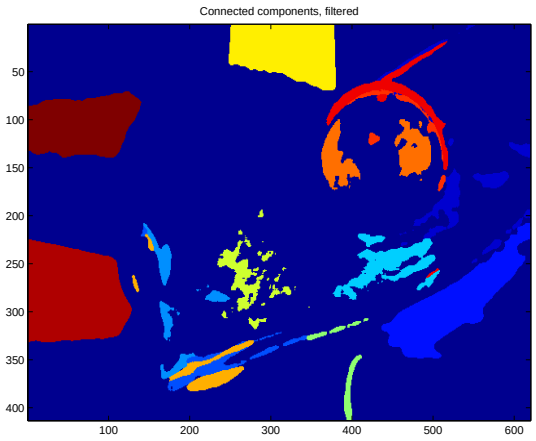

Figure 27: Figure shows the connected component images as before but having replaced the filters using a sliding window technique. In this case the parameters include a region of 7 x 7 pixels that slides across the image replacing the pixels with the most modal pixel in that window.

And now finally we look at the connected component prototypes and the effect on segmentation, later applying the filter to have the smoothening affect removing the rawness of the data.

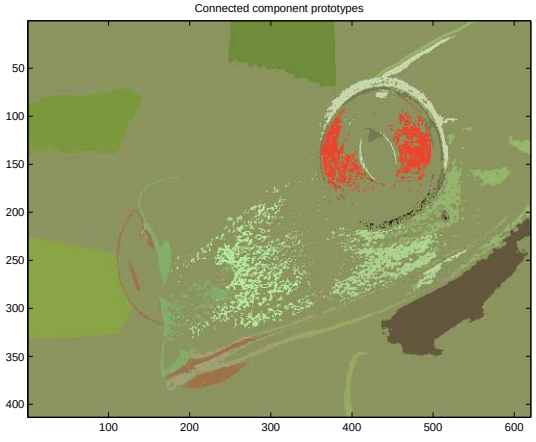

Figure 28: Connected components prototypes solely using the image segmentation techniques. The rawness of the image is pretty visible due to the "grainy" effect.

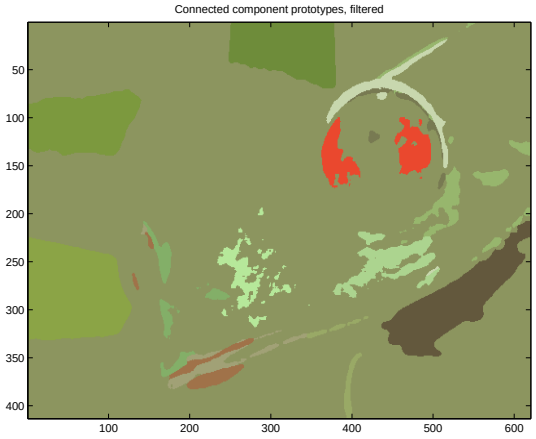

Figure 29: Figure shows the connected component images as before but having replaced the filters using a sliding window technique. In this case the parameters include a region of 7 x 7 pixels that slides across the image replacing the pixels with the most modal pixel in that window.

Having this allows to deduce that the segmentation has been fairly evaluated with the result.

4.2.3 The flower

Now we use another image to test the effect of complete variation of colors on the image. To carry out the experiment we use the rose[3] image to test the segmentation patters

The next image represents the original image of the frog that will be segmented.

Figure 30: Figure shows the connected component images as before but having replaced the filters using a sliding window technique. In this case the parameters include a region of 7 x 7 pixels that slides across the image replacing the pixels with the most modal pixel in that window.

Now we present the model that was generated by the algorithm when the growing neural gas was applied to it.

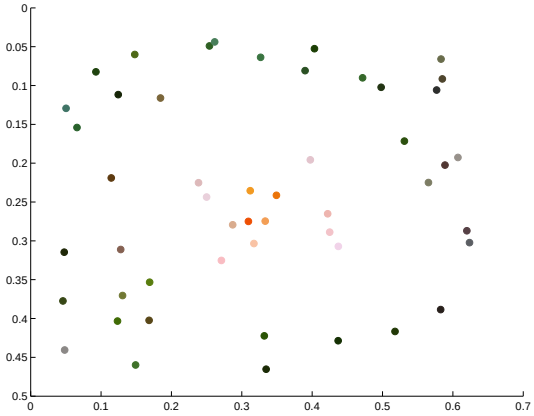

Figure 31: Figure shows the connected component images as before but having replaced the filters using a sliding window technique. In this case the parameters include a region of 7 x 7 pixels that slides across the image replacing the pixels with the most modal pixel in that window.

As it is seen in the photo, having a variety of colors has a much more different effect on the image than the previous one which included shades. Here the edges of the flower are much more visible with the shades showing a much brighter effect than the previous result.

Figure 32: Figure shows the connected component images as before but having replaced the filters using a sliding window technique. In this case the parameters include a region of 7 x 7 pixels that slides across the image replacing the pixels with the most modal pixel in that window.

To create a clearer view of the segmented images we have a look at the connected samples and then we apply the filter to see the difference between the two images.

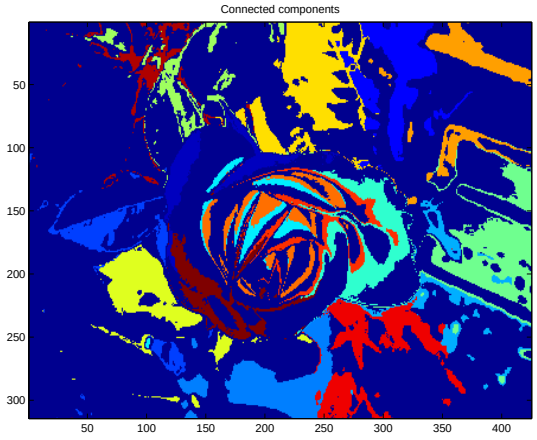

Figure 33: This image shows the connected components of the image where the samples have been replaced by their winner prototypes. This applies the pseudo-colors for a better enhancement of the regions

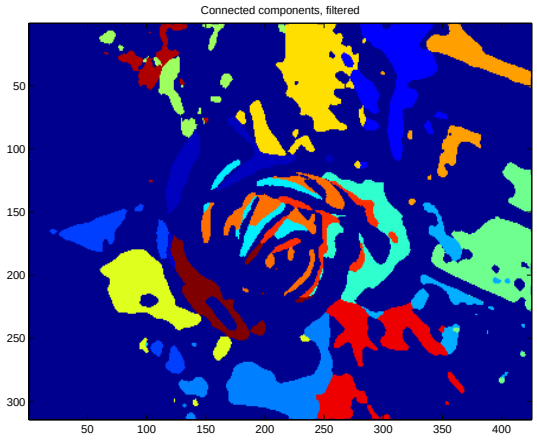

Figure 34: Figure shows the connected component images as before but having replaced the filters using a sliding window technique. In this case the parameters include a region of 7 x 7 pixels that slides across the image replacing the pixels with the most modal pixel in that window.

And now finally we look at the connected component prototypes and the effect on segmentation, later applying the filter to have the smoothening affect removing the rawness of the data.

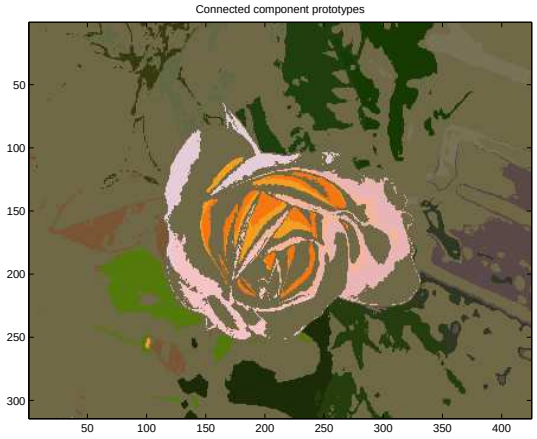

Figure 35: Connected components prototypes solely using the image segmentation techniques. The rawness of the image is pretty visible due to the "grainy" effect.

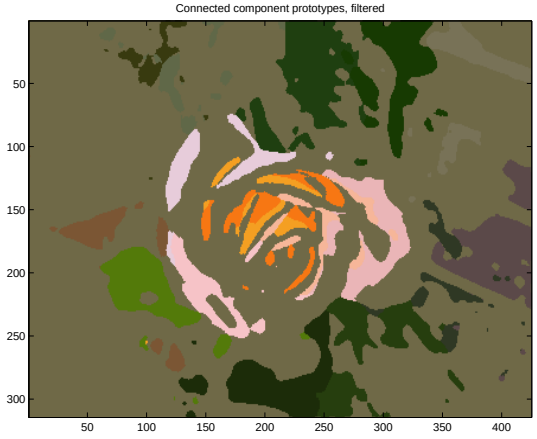

Figure 36: Figure shows the connected component images as before but having replaced the filters using a sliding window technique. In this case the parameters include a region of 7 x 7 pixels that slides across the image replacing the pixels with the most modal pixel in that window.

Having seen three types of images that have been segmented we could look for an image that might be more colorful and contain various colors. For this purposes we choose two contradictory following images that would allow our results to be coherent.

4.2.4 The Apple

Now we use another image to test the effect of simple image now with not may colors and a more realistic effect.

The next image represents the original simple image of the apple[1] that will be segmented.

Figure 37: Figure shows the connected component images as before but having replaced the filters using a sliding window technique. In this case the parameters include a region of 7 x 7 pixels that slides across the image replacing the pixels with the most modal pixel in that window.

Now we present the model that was generated by the algorithm when the growing neural gas was applied to it.

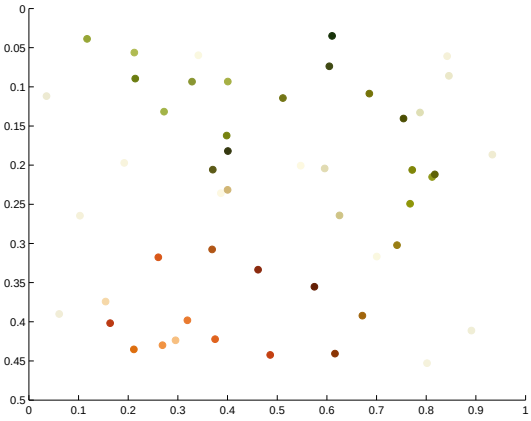

Figure 38: Figure shows the connected component images as before but having replaced the filters using a sliding window technique. In this case the parameters include a region of 7 x 7 pixels that slides across the image replacing the pixels with the most modal pixel in that window.

As it is seen in the photo, in this image there lacks a choice of various colors and therefore the model generated by the program seems to be much simpler in comparison to the previously generated models. This would allow the graph to be trained much as only a few nodes would be able to cover up the graph

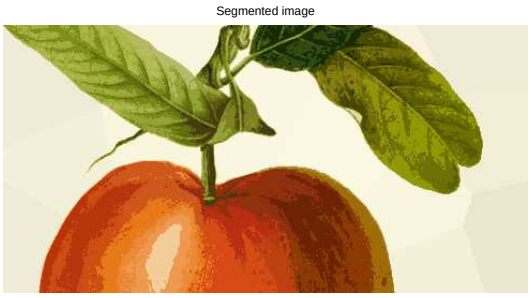

Figure 39: Figure shows the connected component images as before but having replaced the filters using a sliding window technique. In this case the parameters include a region of 7 x 7 pixels that slides across the image replacing the pixels with the most modal pixel in that window.

To create a clearer view of the segmented images we have a look at the connected samples and then we apply the filter to see the difference between the two images.

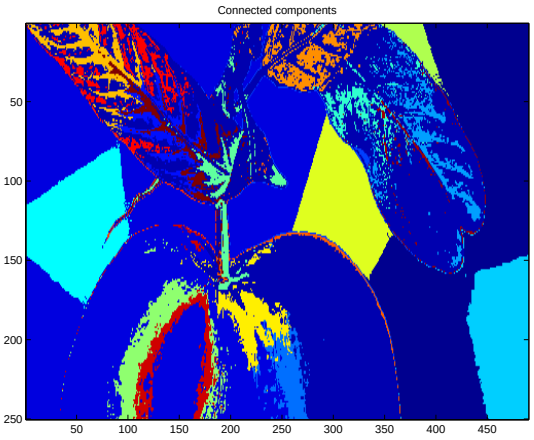

Figure 40: This image shows the connected components of the image where the samples have been replaced by their winner prototypes. This applies the pseudo-colors for a better enhancement of the regions.

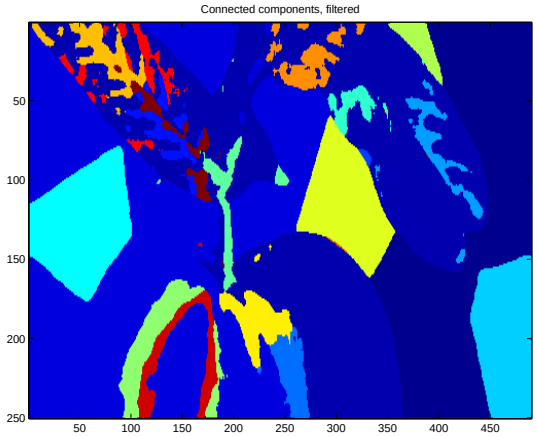

Figure 41: Figure shows the connected component images as before but having replaced the filters using a sliding window technique. In this case the parameters include a region of 7 x 7 pixels that slides across the image replacing the pixels with the most modal pixel in that window.

And now finally we look at the connected component prototypes and the effect on segmentation, later applying the filter to have the smoothening affect removing the rawness of the data.

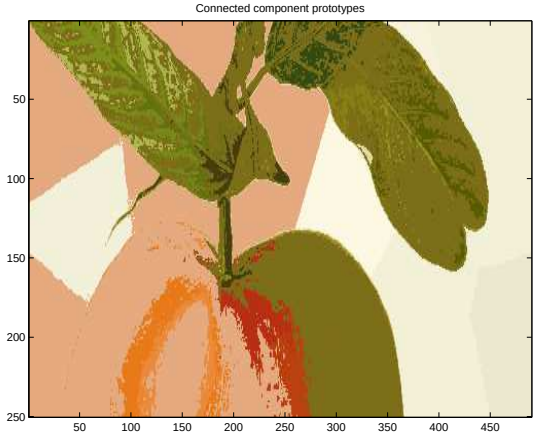

Figure 42: Connected components prototypes solely using the image segmentation techniques. The rawness of the image is pretty visible due to the "grainy" effect.

Figure 43: Figure shows the connected component images as before but having replaced the filters using a sliding window technique. In this case the parameters include a region of 7 x 7 pixels that slides across the image replacing the pixels with the most modal pixel in that window.

As we have seen the model graph did not seem to vary in colors as most of the colors were of a similar share and therefore the smoothening process seemed a bit more complex as the network has not been able to learn the distinctions between the colors effectively.

4.2.5 The Baboon

This image is one with a much more stronger affect on the shades and on the coloring as it contains various disjunctive colors. We pass it through our segmentation script to see the affect the script has to this image.

The next image represents the original simple image of the baboon that will be segmented.

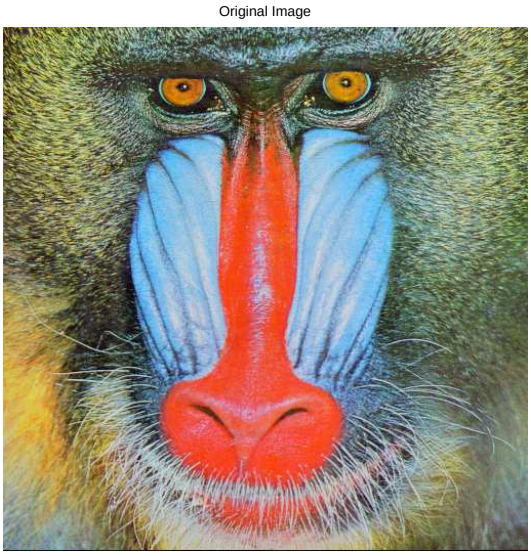

Figure 44: Figure shows the connected component images as before but having replaced the filters using a sliding window technique. In this case the parameters include a region of 7 x 7 pixels that slides across the image replacing the pixels with the most modal pixel in that window.

Now we present the model that was generated by the algorithm when the growing neural gas was applied to it.

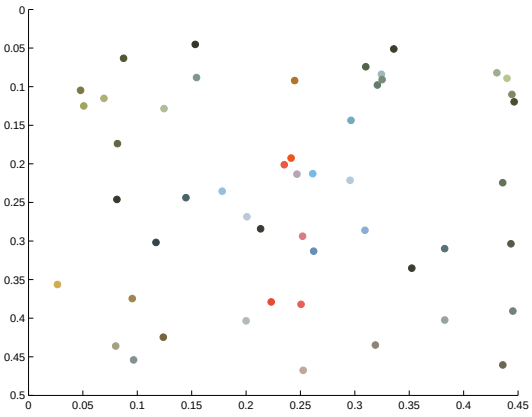

Figure 45: Figure shows the connected component images as before but having replaced the filters using a sliding window technique. In this case the parameters include a region of 7 x 7 pixels that slides across the image replacing the pixels with the most modal pixel in that window.

As it is seen in the photo, in this image there lacks a choice of various colors and therefore the model generated by the program seems to be much simpler in comparison to the previously

generated models. This would allow the graph to be trained much as only a few nodes would be able to cover up the graph

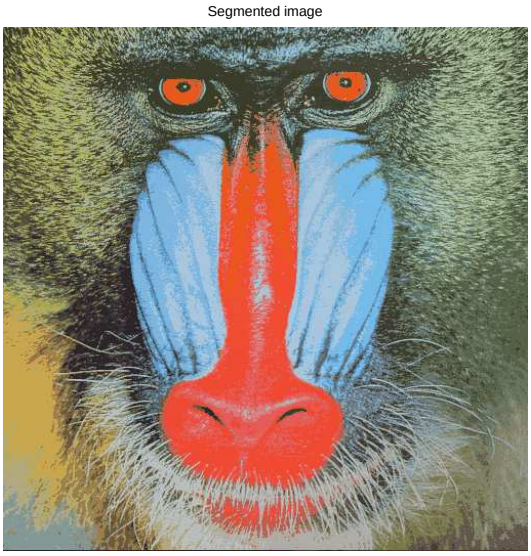

Figure 46: Figure shows the connected component images as before but having replaced the filters using a sliding window technique. In this case the parameters include a region of 7 x 7 pixels that slides across the image replacing the pixels with the most modal pixel in that window.

To create a clearer view of the segmented images we have a look at the connected samples and then we apply the filter to see the difference between the two images.

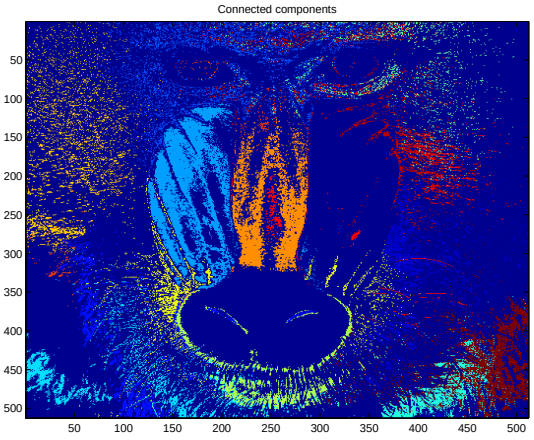

Figure 47: This image shows the connected components of the image where the samples have been replaced by their winner prototypes. This applies the pseudo-colors for a better enhancement of the regions.

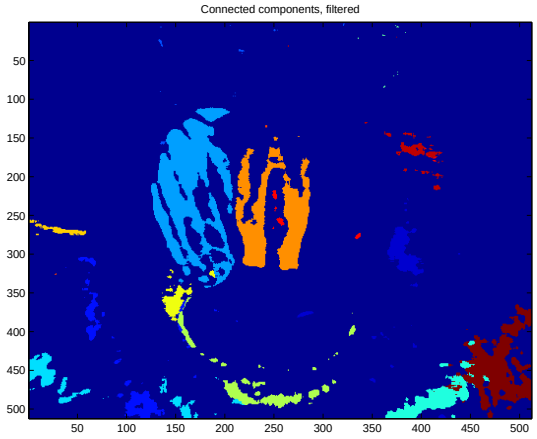

Figure 48: Figure shows the connected component images as before but having replaced the filters using a sliding window technique. In this case the parameters include a region of 7 x 7 pixels that slides across the image replacing the pixels with the most modal pixel in that window.

And now finally we look at the connected component prototypes and the effect on segmentation, later applying the filter to have the smoothening affect removing the rawness of the data.

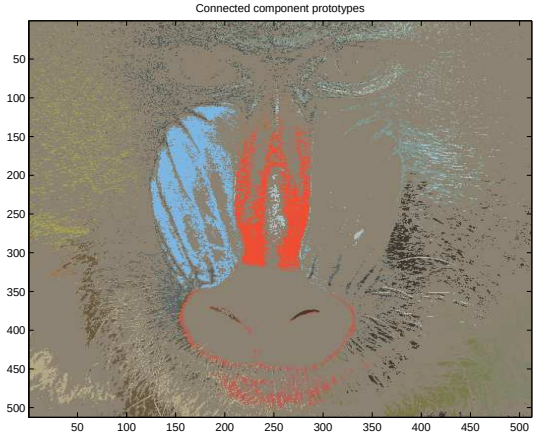

Figure 49: Connected components prototypes solely using the image segmentation techniques. The rawness of the image is pretty visible due to the "grainy" effect.

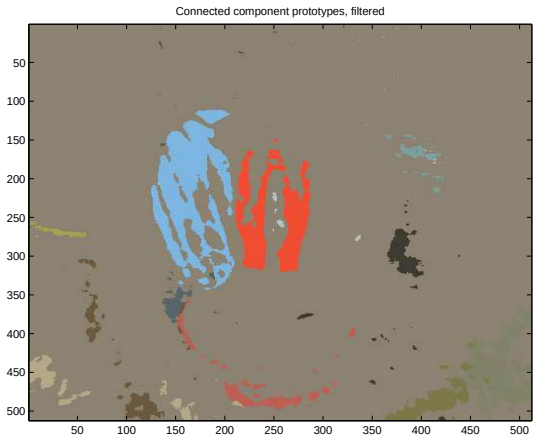

Figure 50: Figure shows the connected component images as before but having replaced the filters using a sliding window technique. In this case the parameters include a region of 7 x 7 pixels that slides across the image replacing the pixels with the most modal pixel in that window.

Here we conclude our experiments with the images to juxtapose the different stages of the algorithm with various images. In many of the occasions it is visible that the change in the algorithm seems to have a strong

5 Conclusions

5.1 English

Having a look at the various results that were produced as a result of execution we can clearly deduce two main factors from the results that have been chosen. One of the factors chosen to be tested was the *NumConnectionsToDelete* which was the caused a specific number of connections to be deleted from a graph. After the having carried out tests where we change the parameter of the number of connections to be deleted from, we obtain a graph with a plot that in has a linear negative correlation. This means that the higher the number of connections that are deleted, the lower the mean squared error is between the input samples and the prototypes. This occurs because the connections that get deleted during the analysis and segmentation of the image produce smaller regions. These smaller regions mean the input samples would be closer to the prototyped samples in the winner regions that actually matter and are used to calculate the minimum squared error. However the decrease in the error would depend on the types of images that are being segmented. The decrease seems linear in the graph and therefore the fewer the number of connections, lower the sum of the mean squared error is between the samples and the prototype however it would only be valid up until a point as deleting more connections than required could result in a distorted segmented image.

The next parameter that was modified for the experimentation purposes was the *SpatialFactor* and we can see from the resulting graphs that there is a decrease in error with the with a higher value of the parameter. This occurrence takes place because of the cohesion between the region. The parameter determines the weight that is given to the coordinates of the pixel, we could deduce that the algorithm learns more when two regions are distinct whenever they are separated by an intermediate region. A higher priority to the coordinates of the graph allows the model to learn more clearly on where the regions are based and therefore allows the correct distinction in the regions whereas a pure RGB combination model would simply classify most of the separated regions with a similar RGB combination as one region. This would be useful if that's the originally planned output however in most cases the position of the region would matter.

We have only tested two parameters in this work whilst keeping the others constant however we could vary the combination of the parameters to see the correlation between the *SpatialFactor* and *NumConnectionsToDelete*. We could also test out various parameters that affect the algorithm training such as *MaxUnits* or even changing the window size of the filtering effect that is carried out by creating bigger regions to smoothen the raw images after the segmentation. Once the segmentation has been carried out we could also use this segmentation as means to recognize daily objects where the neurons would be trained specially to segment different viewing angles of an image and later be used for the detection of objects in the segmented images we obtain from our Growing Neural Gas model. The optimization of the algorithm would play a crucial part and also its improvement in factors, such as the times at which the newer nodes are added to

the algorithm, or even the halting point of the algorithm, would allow the algorithm to run more precisely segmenting the image more accurately. In this modifications the newer nodes could be added when the changes in the re-positioning of the nodes in the graph becomes maximal rather than the usual constant addition of nodes at a given multiple of iterations. The halting point of the algorithm could be determined by a minimum error between the previous and the current positions of the nodes showing the changes in graphs become minimal and therefore further iterations would not result in a better model than the one previously established.

As a final conclusion, it can be said that the newly proposed algorithm has shown a potential to become a new alternative for unsupervised image segmentation. More work is necessary in order to examine its merits.

5.2 Español

Revisados los distintos resultados producidos por los experimentos se puede deducir algunas conclusiones de los factores elegidos. El primero de los parámetros que habíamos elegido era el *NumConnectionsToDelete* que elimina un número específico de las conexiones del modelo. Después de haber ejecutado las pruebas donde cambiamos el parámetro que elimina un número de conexiones del modelo, obtenemos una gráfica donde se nota una correlación lineal negativa. Esto significa, que cuantas mas conexiones se eliminan, menor será el error cuadrático medio entre las entradas y los prototipos. Esto ocurre porque las conexiones que se eliminan durante el proceso de análisis y segmentación de la imagen producen regiones mas pequeñas. Esto produce que las entradas estén mas cerca de los prototipos en las regiones que se usan para calcular el error cuadrático medio. Sin embargo, el decrecimiento del error depende de los tipos de imágenes que elijamos para segmentar. El decrecimiento en este caso es lineal, que significa que con menos conexiones en el modelo, menos error habrá, pero esto sería valido solo hasta un cierto punto, ya que quizás eliminamos mas conexiones de lo necesario que nos daría como salida una imagen distorsionada.

El segundo parámetro que se había modificado para la experimentación es el *SpatialFactor* y podemos ver, como en gráfica antigua, hay un decrecimiento en el error con un mayor valor del parámetro. Esto suele pasar porque las regiones están mas cohesionadas. El parámetro determina la ponderación de las coordenadas del píxel, y podríamos deducir que el algoritmo aprende cada vez mejor si dos regiones se consideran distintas si están separadas por una región intermedia. Dar una mayor prioridad a las coordenadas permitiría al modelo aprender las regiones mejor y por lo tanto podría distinguir correctamente entre las regiones, mientras que en un modelo solo RGB se clasificaría todas las regiones del mismo color como la misma en vez de separadas, aunque estén situadas en lugares distintos de la imagen.

Hemos estudiado solo dos parámetros en este trabajo manteniendo los otros parámetros. También podríamos cambiar estos parámetros para ver si tienen alguna correlación. Hay mas parámetros como el *MaxUnits* que podríamos usar para ver como afecta el cambio de ese parámetro al algoritmo, o también cambiando el tamaño de las ventanas de filtrado para usar regiones mas

grandes o pequeñas viendo el efecto que tiene esto sobre el suavizado de las imágenes. Una vez realizamos la segmentación podríamos usar esto para determinar y detectar objetos o formas en las imágenes, grabando distintos ángulos del mismo objeto. La optimización del algoritmo del Gas Neuronal Creciente también es un factor importante, que se podría mejorar probando varios factores, como por ejemplo, añadir los nodos dinámicamente o simplemente diciendo cuando se para el entrenamiento del modelo. En estas modificaciones, por ejemplo en la modificación de añadir nodos nuevos, se podría añadir un nodo cuanto se nota que hay cambios grandes en los re-posicionamientos de los nodos existentes. La parada del entrenamiento podría estar determinada por un error mínimo en los re-posicionamientos de los nodos del modelo, que significa que la diferencia entre los posiciones anteriores de los nodos y las posiciones existentes no sea menor que un mínimo predeterminado, ya que las iteraciones no cambiarían el modelo a partir de este punto.

Como conclusión final podemos decir que el nuevo algoritmo propuesto ha mostrado el potencial de ser una nueva alternativa para la segmentación de imágenes no supervisada. Aún es necesario profundizar sobre el asunto para examinar sus méritos.

References

[1] Image of an apple. URL http://www.telegraph.co.uk/news/earth/earthpicturegalleries/8504667/Colourful-tree-frogs-photographed-by-Angi-Nelson.html?image=6.

[2] Face Recognition. URL http://code.opencv.org/attachments/950/eigenfaces_opencv.png.

[3] Rosebud flower. URL http://images.all-free-download.com/images/graphicthumb/rose_rosebud_flower_221924.jpg.

[4] ANGI NELSON/HOTSPOT MEDIA. Colourful tree frogs photographed by Angi Nelson. URL http://www.telegraph.co.uk/news/earth/earthpicturegalleries/8504667/Colourful-tree-frogs-photographed-by-Angi-Nelson.html?image=6.

[5] Frtizke B. A growing neural gas network learns topologies. *MIT Press*, 1991.

[6] Changhyun Choi, BulletAlexander J. B. Trevor, BulletHenrik I. Christensen. RGB-D Edge Detection and Edge-Based Registration, 2013. URL http://www.cc.gatech.edu/~cchoi/rgbd_edges.html.

[7] G. Papandreou and P. Maragos. Multigrid Geometric Active Contour Models, 2007. URL http://cvsp.cs.ntua.gr/research/gac/.

[8] Jaume Bacardit. Biological Data Mining. URL http://www.cs.nott.ac.uk/~jqb/G53BIO/Slides/Biological%20Data%20Mining.ppt.

[9] National Instruments Corporation. Morphological segmentation, 2013. URL http://zone.ni.com/reference/en-XX/help/372916P-01/nivisionconcepts/morphological_segmentation/.

[10] NCI-MICCAI 2013 Grand Challenges in Image Segmentation. Brain image segmentation for the detection of tumors, 2013. URL http://martinos.org/qtim/miccai2013/.

[11] Tolga Birdal. Image Processing Basics in C#, 2008. URL http://www.codeproject.com/Articles/31014/Image-Processing-Basics-in-C.

[12] Vighnesh Birodkar. Graph based Image Segmentation, 2014. URL https://vcansimplify.wordpress.com/2014/05/16/graph-based-image-segmentation/.

[13] Wikia. Artificial neural network . URL http://psychology.wikia.com/wiki/Artificial_neural_network.