*People's Democratic Republic of Algeria*
*Ministry of Higher Education and Scientific Research*

***UNIVERSITY OF Dr. TAHAR MOULAY SAIDA***
**FACULTY OF TECHNOLOGY**
**DEPARTMENT OF COMPUTER SCIENCE**


# ***Master's thesis***

**SPECIALIZATION: Computer Network and Distributed Systems**

## ***Theme***

# Parallel implementation of Kohonen maps for color image clustering

**Presented by:**
**Ababou Abdelkader**

**Supervisor:**
**Mr. Djelloul Mokadem**

**September 2018**

**Abstract**

Artificial neural networks (ANN) represent a universal model, which can be leveraged to solve a great variety of tasks. The latest research showed a significant progress of this field in the past few decades. After the parallel hardware became available for reasonable prices, it fueled the research of efficient optimization of ANN by leveraging parallel architectures. In this thesis , We present parallel implementation of Kohonens Self-Organizing Map (SOM) neural network for segmentation of color images.The efficiency of the implemented confirmed by the results of the performed tests.
**Key words:** Self-Organizing Map,neural networks , Color images segmentation , Parallel learning algorithms , Data Mining , Image clustering .

# Acknowledgement

I had the great pleasure of working with my supervisors Mr Djelloul Mokadem, I would like to thank him for all support and encouragement during the whole project. Without your help this would have been an entirely different project .

I wish to devote a special vote of thanks to my parent and family for their unconditional love. I would always be indebted to them for believing in me and my decisions. This work would not have been remotely possible without their support.

# Contents

# Chapter 1

# Introduction

Image segmentation is the technique of dividing or partitioning an image into parts, called segments. Applications of image segmentation include Content-Based image Retrieval (CBIR), image compression, object detection , object recognition , medical imaging.

The Content-based image retrieval systems search the repository of images based on features of the digital images rather than labels. Major features include texture, color, object structure or any other selective information that can be extracted from the image itself. applications of CBIR systems include, expert medical systems, military use, remote sensing systems. The main bottleneck of CBIR systems is the inability to process high dimensional data . In a color image, where 8 bits for the representation of each color component (R, G, and B) are used, there are 16 million possible colors in that image. We need to decrease these dimensions in order to get meaningful results from the CBIR systems. Image segmentation is a method that helps decrease the number of colors required to represent an image by dividing an image into regions and by assigning each region with the same color. Previous works have employed several techniques edge-detection , region growing, histogram-based, graph partitioning were used for image segmentation.[1]

The Human Visual System (HVS) has tremendous ability to segment and extract information from the images. This remarkable ability of the HVS is obviously a great motivation for anyone to apply Artificial Neural Networks in field of image segmentation. ANNs have many advantages such as, massive parallelism, fault-tolerance to missing, noisy or outlier attributes, better adaptability on different datasets and optimal or near optimal performance [1]. For image segmentation, mainly three types of ANNs are used: su-

pervised, unsupervised, and a combination of the both. When segmenting color images, ANNs with unsupervised learning are preferred over the others. Because the former needs training samples for training and in some cases, training data might not be available at all.

Self-Organizing Map (SOM) Neural Network is a type of unsupervised ANN. It has two major characteristics, it reduces the dimensions of data it groups together similar samples. These two characteristics of SOM help us in segmenting the regions of the image that has similar features, and it reduces the number of colors required to represent an image.

The Self-Organizing-Map which are also used in domains like speech recognition and data classification is very computationally expensive in training process.the aime of this project is effective parallel implementation of SOM for segmentation of color images , to reduce the time taken in training and enhance the performance of Self-Organizing Maps.

## 1.1 How this thesis is organized

Chapter 2, draws the background and motivation for this work .
In chapter 3 , the theory used in the project is presented ,It begins with general introduction to ANN Thereafter, an overview is made about image processing ,Then it proceeds with general discussion about SOM and focusing on training SOM, parallel computing also is presented.
Chapter 4 , Contains a description of how it was implemented and the Testing of the implementation, the result are also presented.

# Chapter 2

# Background

While not strictly part of the project, the topics in this chapter should give a general idea about other image segmentation methods .

## 2.1 Introduction

Digital Image Processing consists of several steps.The first step is image acquisition-that is, to acquire a digital image. After a digital image has been obtained, the next step deals with preprocessing that image. The key function of preprocessing is to improve the image in ways that increase the chances for success of the other processes. The next stage deals with image segmentation. Image segmentation partitions an input image into its constituent parts or objects.
The next step is representation and description. Representation is the transformation of raw data into a descriptive form suitable for computer processing. Description deals with extracting features that result in some quantitative information of interest. Such descriptions are necessarily task specific. The last step is recognition and interpretation. Recognition is the process that assigns a label to an object based on the information of the object. Interpretation assigns meaning to recognized objects .

Among all the senses of perception that we possess, vision is undebatably the most important. We are capable of extracting a wealth of information from an image, which can range from finding objects while we are walking

across a room to detect abnormalities in a medical image. Moreover, things as simple as catching a ball which is coming towards us requires to extract an incredible amount of information in a small portion of time: we need to recognise the ball, track its movement, measure its position and distance, estimate its trajectory... and it is only a childs game! The subconscious way that we often look, interpret and ultimately act upon what we see, belies the real complexity and effectiveness of the Human Visual System. The comparatively young science of Computer Vision tries to emulate the vision system by means of an image capture equipment in place of our eyes, and computer and algorithms in place of our brain. More formally, Computer Vision can be defined as the process of extracting relevant information of the physical world from images using a computer to obtain this information . The final goal would be to develop a system that could understand a picture in much the same way that a human

observer can. Nevertheless, the great complexity of the Human Visual System makes this aim to be regarded for the moment only as an utopian wish, and current systems try to solve more basic and specific problems.

One of the basic abilities of the Human Visual System is the capability of grouping the image into a set of regions which contain pixels with some common characteristic. This task is usually referred as segmenting the image. The common characteristic used as basis for the segmentation can be a simple pixel property such as grey level or colour. However, an image can also be segmented according to a more complex non-local property such as texture. Some examples of image segmentation are shown Figure 2.1
The goal of image segmentation is to detect and extract the regions which compose an image. Note that contrary to the classification problem, recognition of these regions is not required. In fact, although it is difficult to conceive, we can think in image segmentation as the first look that we made at the world when we were newborn. In other words, to look without higher knowledge about the objects that we can see in the scene. Hence, it is not possible to identify the different objects, simply because it is the first time that we see them. So, the answer of the image segmentation process will be something as: there are four regions in the image and an array of the size of the image where each pixel is labelled with the corresponding region number. Two of the basic approaches for image segmentation are region and boundary based. The literature for the last 30 years is full of a large set of propos-

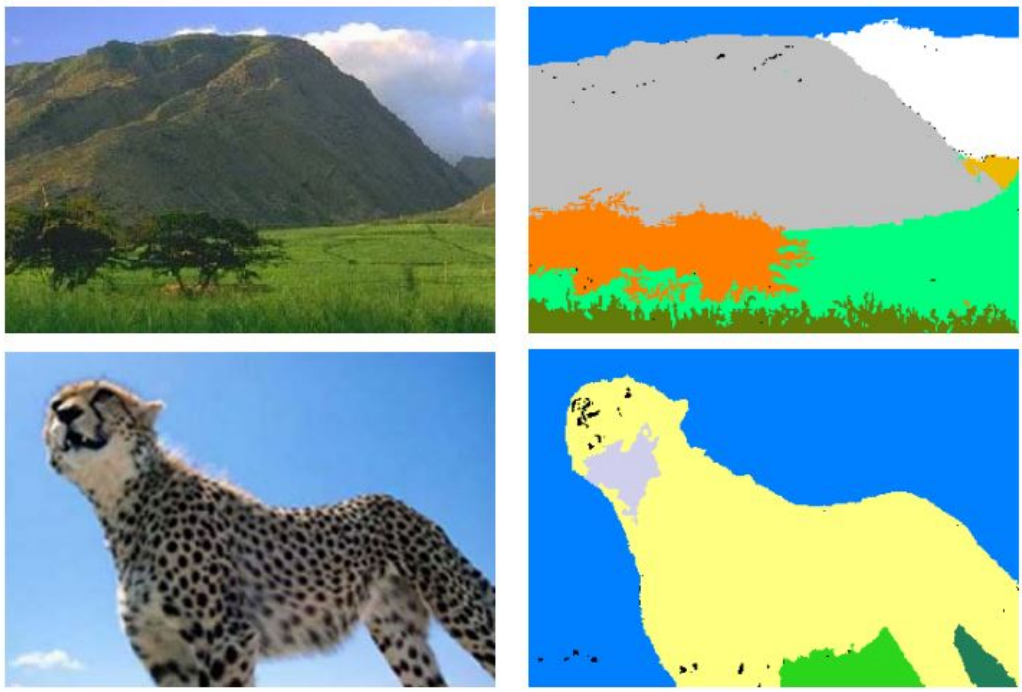



Figure 2.1: Image segmentation. The original images in column (a) are partitioned into their meaningful regions, which are visually distinct and uniform, in the segmented images (b).

als which attempt to segment the image based on one of these approaches. However, based on the complementary nature of edge and region information, current trends on image segmentation wage for the integration of both sources in order to obtain better results and to solve the problems that both methods bear when are used separately. There are also two basic properties that can be considered for grouping pixels and define the concept of similarity that would form regions: colour and texture. The importance of both features in order to define the visual perception is obvious in images corresponding to natural scenes, which have considerable variety of colour and texture. However, most of the literature deals with segmentation based on either colour or texture, and few proposals consider both properties together. Fortunately, this tendency seems to be changing in the actuality originated

by the intuition that using information provided by both features, one should be able to obtain more robust and meaningful results.

## 2.2 Methods

### 2.2.1 Thresholding

The simplest method of image segmentation is called the thresholding method. This method is based on a clip-level (or a threshold value) to turn a gray-scale image into a binary image. There is also a balanced histogram thresholding.

The key of this method is to select the threshold value (or values when multiple-levels are selected). Several popular methods are used in industry including the maximum entropy method, Otsu's method (maximum variance), and k-means clustering.

Recently, methods have been developed for thresholding computed tomography (CT) images. The key idea is that, unlike Otsu's method, the thresholds are derived from the radiographs instead of the (reconstructed) image.

New methods suggested the usage of multi-dimensional fuzzy rule-based non-linear thresholds. In these works decision over each pixel's membership to a segment is based on multi-dimensional rules derived from fuzzy logic and evolutionary algorithms based on image lighting environment and application.[16]

### 2.2.2 Clustering methods

Clustering is a process whereby a data set is replaced by clusters, which are collections of data points that belong together. It is natural to think of image segmentation as clustering, grouping those pixels that have the same colour and/or the same texture. Clustering methods can be divided into two basic types: hierarchical and partitional clustering. Within each of the types there exists a wealth of subtypes and different algorithms for finding the clusters. Hierarchical clustering proceeds successively by either merging smaller clus-

ters into larger ones (agglomerative algorithms), or by splitting larger clusters (divisive algorithms). The clustering methods differ in the rule by which it is decided which two small clusters are merged or which large cluster is split. The final result of the algorithm is a tree of clusters called a dendogram, which shows how the clusters are related. By cutting the dendogram at a desired level, a clustering of the data items into disjoint groups is obtained. On the other hand, partitional clustering attempts to directly decompose the data set into a set of disjoint clusters. An objective function expresses how good a representation is, and then the clustering algorithm tries to minimize this function in order to obtain the best representation. The criterion function may emphasize the local structure of the data, as by assigning clusters to peaks in the probability density function, or the global structure. Typically the global criteria involves minimizing a measure of dissimilarity for the samples within each cluster, while maximizing the dissimilarity between different clusters.
The most commonly used partitional clustering method is the K-means algorithm , in which the criterion function is the squared distance of the data items from their nearest cluster centroids.

Clustering methods, even as thresholding methods, are global and do not retain positional information. The major drawback of this is that it is invariant to spatial rearrangement of the pixels, which is an important aspect of what is meant by segmentation. Resulting segments are not connected and can be widely scattered. Some attempts have been made to introduce such information using pixels coordinates as features. However, this approach tends to result in large regions being broken up and the results so far are no better than those that do not use spatial information . The need to incorporate some form of spatial information into the segmentation process, led to the development of methods where pixels are classified using their context or neighbourhood.[15]

### 2.2.3 Compression-based methods

Compression based methods postulate that the optimal segmentation is the one that minimizes, over all possible segmentations, the coding length of the data. The connection between these two concepts is that segmentation tries to find patterns in an image and any regularity in the image can be used to compress it. The method describes each segment by its texture and

boundary shape. Each of these components is modeled by a probability distribution function and its coding length is computed as follows:

1. The boundary encoding leverages the fact that regions in natural images tend to have a smooth contour. This prior is used by Huffman coding to encode the difference chain code of the contours in an image. Thus, the smoother a boundary is, the shorter coding length it attains.

2. Texture is encoded by lossy compression in a way similar to minimum description length (MDL) principle, but here the length of the data given the model is approximated by the number of samples times the entropy of the model. The texture in each region is modeled by a multivariate normal distribution whose entropy has a closed form expression. An interesting property of this model is that the estimated entropy bounds the true entropy of the data from above. This is because among all distributions with a given mean and covariance, normal distribution has the largest entropy. Thus, the true coding length cannot be more than what the algorithm tries to minimize.

For any given segmentation of an image, this scheme yields the number of bits required to encode that image based on the given segmentation. Thus, among all possible segmentations of an image, the goal is to find the segmentation which produces the shortest coding length. This can be achieved by a simple agglomerative clustering method. The distortion in the lossy compression determines the coarseness of the segmentation and its optimal value may differ for each image. This parameter can be estimated heuristically from the contrast of textures in an image. For example, when the textures in an image are similar, such as in camouflage images, stronger sensitivity and thus lower quantization is required[16].

### 2.2.4 Histogram-based methods

Histogram-based methods are very efficient compared to other image segmentation methods because they typically require only one pass through the pixels. In this technique, a histogram is computed from all of the pixels in the image, and the peaks and valleys in the histogram are used to locate the clusters in the image. Color or intensity can be used as the measure.

A refinement of this technique is to recursively apply the histogram-seeking method to clusters in the image in order to divide them into smaller clusters. This operation is repeated with smaller and smaller clusters until no more clusters are formed.

One disadvantage of the histogram-seeking method is that it may be difficult to identify significant peaks and valleys in the image.

Histogram-based approaches can also be quickly adapted to apply to multiple frames, while maintaining their single pass efficiency. The histogram can be done in multiple fashions when multiple frames are considered. The same approach that is taken with one frame can be applied to multiple, and after the results are merged, peaks and valleys that were previously difficult to identify are more likely to be distinguishable. The histogram can also be applied on a per-pixel basis where the resulting information is used to determine the most frequent color for the pixel location. This approach segments based on active objects and a static environment, resulting in a different type of segmentation useful in video tracking[16].

### 2.2.5 Edge detection

Edge detection is a well-developed field on its own within image processing. Region boundaries and edges are closely related, since there is often a sharp adjustment in intensity at the region boundaries. Edge detection techniques have therefore been used as the base of another segmentation technique.

The edges identified by edge detection are often disconnected. To segment an object from an image however, one needs closed region boundaries. The desired edges are the boundaries between such objects or spatial-taxons.

Spatial-taxons are information granules, consisting of a crisp pixel region, stationed at abstraction levels within a hierarchical nested scene architecture. They are similar to the Gestalt psychological designation of figure-ground, but are extended to include foreground, object groups, objects and salient object parts. Edge detection methods can be applied to the spatial-taxon region, in the same manner they would be applied to a silhouette. This method is particularly useful when the disconnected edge is part of an illusory contour

Segmentation methods can also be applied to edges obtained from edge detectors. Lindeberg and Li developed an integrated method that segments edges into straight and curved edge segments for parts-based object recognition, based on a minimum description length (MDL) criterion that was

optimized by a split-and-merge-like method with candidate breakpoints obtained from complementary junction cues to obtain more likely points at which to consider partitions into different segments[16].

### 2.2.6 Region-growing methods

Region-growing methods rely mainly on the assumption that the neighboring pixels within one region have similar values. The common procedure is to compare one pixel with its neighbors. If a similarity criterion is satisfied, the pixel can be set to belong to the cluster as one or more of its neighbors. The selection of the similarity criterion is significant and the results are influenced by noise in all instances.

The method of Statistical Region Merging (SRM) starts by building the graph of pixels using 4-connectedness with edges weighted by the absolute value of the intensity difference. Initially each pixel forms a single pixel region. SRM then sorts those edges in a priority queue and decide whether or not to merge the current regions belonging to the edge pixels using a statistical predicate.

One region-growing method is the seeded region growing method. This method takes a set of seeds as input along with the image. The seeds mark each of the objects to be segmented. The regions are iteratively grown by comparison of all unallocated neighboring pixels to the regions. The difference between a pixel's intensity value and the region's mean, $\delta$ , is used as a measure of similarity. The pixel with the smallest difference measured in this way is assigned to the respective region. This process continues until all pixels are assigned to a region. Because seeded region growing requires seeds as additional input, the segmentation results are dependent on the choice of seeds, and noise in the image can cause the seeds to be poorly placed.

Another region-growing method is the unseeded region growing method. It is a modified algorithm that does not require explicit seeds. It starts with a single region $A_1$ the pixel chosen here does not markedly influence the final segmentation. At each iteration it considers the neighboring pixels in the same way as seeded region growing. It differs from seeded region growing in that if the minimum $\delta$ is less than a predefined threshold $T$ then it is added to the respective region $A_j$. If not, then the pixel is considered different from all current regions $A_i$ and a new region $A_{n+1}$ is created with this pixel.

One variant of this technique, proposed by Haralick and Shapiro (1985), is based on pixel intensities. The mean and scatter of the region and the

intensity of the candidate pixel are used to compute a test statistic. If the test statistic is sufficiently small, the pixel is added to the region, and the regions mean and scatter are recomputed. Otherwise, the pixel is rejected, and is used to form a new region.

A special region-growing method is called $\lambda$ -connected segmentation . It is based on pixel intensities and neighborhood-linking paths. A degree of connectivity (connectedness) is calculated based on a path that is formed by pixels. For a certain value of $\lambda$, two pixels are called $\lambda$-connected if there is a path linking those two pixels and the connectedness of this path is at least $\lambda$. $\lambda$-connectedness is an equivalence relation.Split-and-merge segmentation is based on a quadtree partition of an image. It is sometimes called quadtree segmentation.

This method starts at the root of the tree that represents the whole image. If it is found non-uniform (not homogeneous), then it is split into four child squares (the splitting process), and so on. If, in contrast, four child squares are homogeneous, they are merged as several connected components (the merging process). The node in the tree is a segmented node. This process continues recursively until no further splits or merges are possible. When a special data structure is involved in the implementation of the algorithm of the method, its time complexity can reach $O(n \log n)$ , an optimal algorithm of the method[16].

# Chapter 3

# Theory

In this chapter the general theory used in this master thesis project. This is intended to act as a reference for the later chapters where the theory is used.

## 3.1 Artificial Neural Networks

### 3.1.1 Overview

Artificial Neural Networks are the computational models inspired by the human brain. Many of the recent advancements have been made in the field of Artificial Intelligence, including Voice Recognition, Image Recognition, Robotics using Artificial Neural Networks.
These biological methods of computing are considered to be the next major advancement in the Computing Industry.

### 3.1.2 Modeling one neuron

The area of Neural Networks has originally been primarily inspired by the goal of modeling biological neural systems, but has since diverged and become a matter of engineering and achieving good results in Machine Learning tasks. Nonetheless, we begin our discussion with a very brief and high-level description of the biological system that a large portion of this area has been inspired by.

### 3.1.3 Biological motivation

The basic computational unit of the brain is a neuron. Approximately 86 billion neurons can be found in the human nervous system and they are connected with approximately $10^{14} - 10^{15}$ synapses.[2] The diagram below shows a biological neuron (left) and a common mathematical model (right). Each neuron receives input signals from its dendrites. . The combination of dendrites is often referred to as a dendritic tree, which receives excitatory or inhibitory signals from other neurons via an electrochemical exchange of neurotransmitters.
The magnitude of the input signals reach the cell nucleus depends both on the amplitude of the action potentials propagating from the previous neuron and on the conductivity of the ion channels feeding into the dendrites. The ion channels are responsible for the flow of electrical signals passing through the neurons membrane.
More frequent or larger magnitude input signals generally result in better conductivity ion channels, or easier signal propagation. Depending on this signal aggregated from all synapses from the dendritic tree, the neuron is either activated or inhibited after a process called neural summation. The neuron has an electrochemical threshold, analogous to an activation function in artificial neural networks, which governs whether the accumulated information is enough to activate the neuron.

In biological neural networks like the human brain, learning is achieved by making small tweaks to an existing representation its configuration contains significant information before any learning is conducted. The strengths of connections between neurons, or weights, do not start as random, This initial state is, in part, genetically derived, and is the byproduct of evolution. Over time, the network learns how to perform new functions by adjusting both topology and weights. The fact that there is an initial representation that works well for many tasks is supported by research, which suggests that as young as one month old newborns are able to recognize faces demonstrated by their learning to differentiate between strangers and their parents. other words, the concept of a human face has largely been passed down genetically from parent to child.[3]

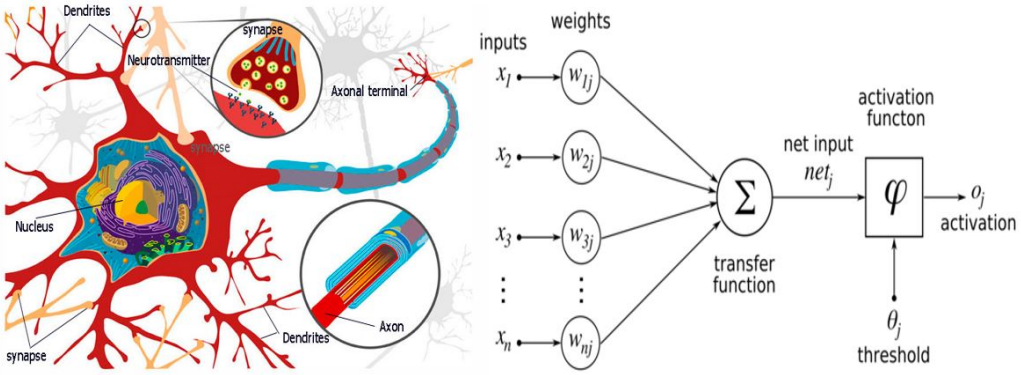



Figure 3.1: biological neuron (left) and its mathematical model (right).

### 3.1.4 How Does Artificial Neural Network Works?

Artificial Neural Networks can be viewed as weighted directed graphs in which artificial neurons are nodes, and directed edges with weights are connections between neuron outputs and neuron inputs.
The Artificial Neural Network receives information from the external world in the form of pattern and image in vector form. These inputs are mathematically designated by the notation $x(n)$ for n number of inputs.
Each input is multiplied by its corresponding weights. Weights are the information used by the neural network to solve a problem. Typically weight represents the strength of the interconnection between neurons inside the Neural Network.
The weighted inputs are all summed up inside computing unit (artificial neuron). In case the weighted sum is zero, bias is added to make the output not-zero or to scale up the system response. Bias has the weight and input always equal to 1.
The sum corresponds to any numerical value ranging from 0 to infinity. To limit the response to arrive at the desired value, the threshold value is set up. For this, the sum is passed through activation function.
The activation function is set to the transfer function used to get the desired output. There are linear as well as the nonlinear activation function.

### Commonly used activation functions

**The binary function** Every activation function takes a single number and performs a certain fixed mathematical operation on it. There are several activation func gtions , Some of the commonly used activation function :

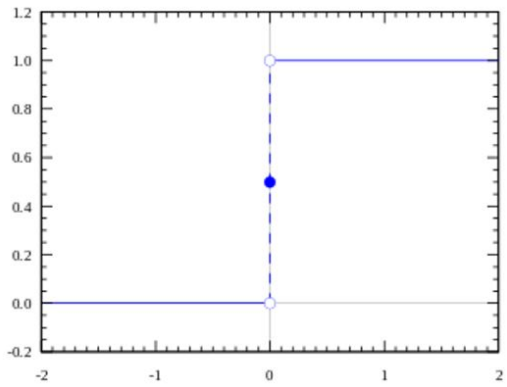



Figure 3.2: binary step activation function .

The simplest activation function is the binary or threshold function which can only take on two value either 1 or 0 , If the input is above a certain Threshold an output is assumed one ( it activated ) otherwise zer $f(x) = \begin{cases} 1 & \text{if } x >= 0 \\ 0 & \text{if } x < 0 \end{cases}$

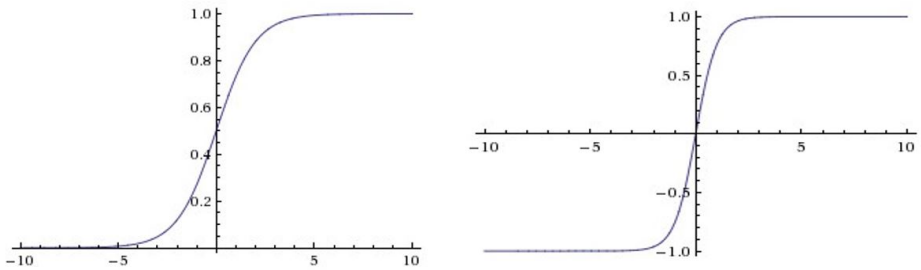



Figure 3.3: Left: Sigmoid non-linearity squashes real numbers to range between [0,1] Right: The tanh non-linearity squashes real numbers to range between [-1,1].

**Sigmoid** The sigmoid non-linearity has the mathematical form $\sigma(x) = \frac{1}{1+e^{-x}}$ and is shown in the image above on the left. it takes a real-valued number and squashes it into range between 0 and 1. In particular, large negative numbers become 0 and large positive numbers become 1. The sigmoid function has seen frequent use historically since it has a nice interpretation as the firing rate of a neuron: from not firing at all (0) to fully-saturated firing at an assumed maximum frequency (1). In practice, the sigmoid non-linearity has recently fallen out of favor and it is rarely ever used.

**Tanh** The tanh non-linearity is shown on the image above on the right. It squashes a real-valued number to the range [-1, 1]. Like the sigmoid neuron, its activations saturate, but unlike the sigmoid neuron its output is zero-centered. Therefore, in practice the tanh non-linearity is always preferred to the sigmoid nonlinearity. Also note that the tanh neuron is simply a scaled sigmoid neuron, in particular the following holds:The tanh non-linearity is shown on the image above on the right. It squashes a real-valued number to the range [-1, 1]. Like the sigmoid neuron, its activations saturate, but unlike the sigmoid neuron its output is zero-centered. Therefore, in practice the tanh non-linearity is always preferred to the sigmoid nonlinearity. Also note that the tanh neuron is simply a scaled sigmoid neuron, in particular the following holds: $tanh(x) = \frac{2}{1+e^{-2x}} - 1$

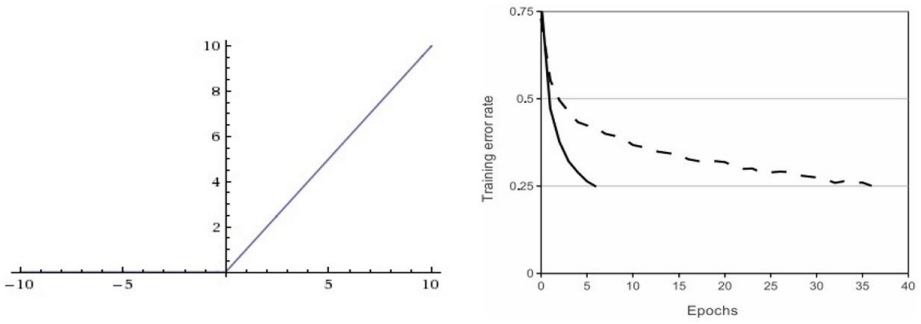



Figure 3.4: left: rectified linear unit (relu) activation function, which is zero when x ¡ 0 and then linear with slope 1 when x ¿ 0. right: a plot from krizhevsky et al. (pdf) paper indicating the 6x improvement in convergence with the relu unit compared to the tan

**ReLu** The Rectified Linear Unit has become very popular in the last few years. It computes function $f(x) = max(0, x)$ In other words, the activation is simply thresholded at zero (see image above on the left).

- (+) It was found to greatly accelerate (e.g. a factor of 6 in Krizhevsky et al.) the convergence of stochastic gradient descent compared to the sigmoid/tanh functions. It is argued that this is due to its linear, non-saturating form.
- (+) Compared to tanh/sigmoid neurons that involve expensive operations (exponentials, etc.), the ReLU can be implemented by simply thresholding a matrix of activations at zero.

### 3.1.5 Artificial Neural Network Architecture

Neural Networks as neurons in graphs. Neural Networks are modeled as collections of neurons that are connected in an acyclic graph. In other words, the outputs of some neurons can become inputs to other neurons. Cycles are not allowed since that would imply an infinite loop in the forward pass of a network. Instead of an amorphous blobs of connected neurons, Neural Network models are often organized into distinct layers of neurons. the most common layer type is the fully-connected layer in which neurons between two adjacent layers are fully pairwise connected. In general, an artificial neural

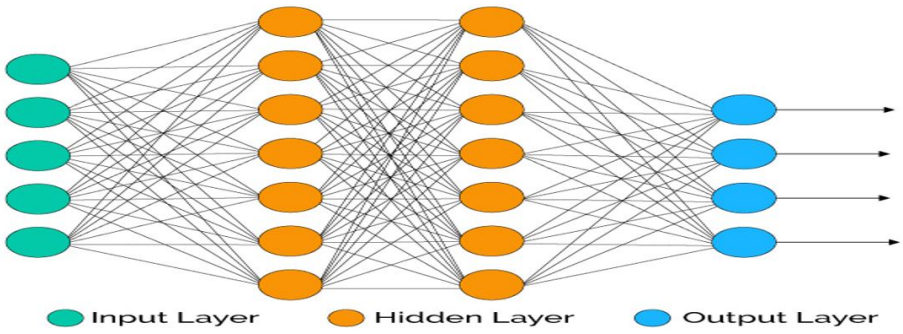



Figure 3.5: fully-connected layer with 3-layer neural network and three inputs, two hidden layers of 7 neurons each and one output layer

network can be divided into three parts, named layers, which are known as:

**Input layer** This layer is responsible for receiving information (data), signals, features, or measurements from the external environment.

**Hidden layers** These layers are composed of neurons which are responsible for extracting patterns associated with the process or system being analyzed. These layers perform most of the internal processing from a network.

**Output layer** This layer is also composed of neurons, and thus is responsible for producing and presenting the final network outputs, which result from the processing performed by the neurons in the previous layers.

### 3.1.6 Learning Techniques and Algorithms in Artificial Neural Networks

The neural network learns by adjusting its weights and bias (threshold) iteratively to yield desired output. These are also called free parameters. For learning to take place, the Neural Network is trained first. The training is performed using a defined set of rules also known as the learning algorithm.

**Training Algorithms For Artificial Neural Networks:**

**Gradient Descent Algorithm:** This is the simplest training algorithm used in case of supervised training model. In case, the actual output is different from target output, the difference or error is find out. The gradient descent algorithm changes the weights of the network in such a manner to minimize this mistake.

**Back Propagation Algorithm:** It is an extension of gradient based delta learning rule. Here, after finding an error (the difference between desired and target), the error is propagated backward from output layer to the input layer via hidden layer. It is used in case of Multilayer Neural Network.

**Other 4 Algorithms to Train a Neural Network:**

- Hebb Rule
- Self - Organizing Kohonen Rule (the subject of this thesis)

- Hopfield law
- LMS algorithm (Least Mean Square)
- Competitive Learning

**Learning Techniques in Neural Networks**

**Supervised Learning** Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output Y = f(X).
The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data. It is called supervised learning because the process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. We know the correct answers, the algorithm iteratively makes predictions on the training data and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance.

**Unsupervised Learning** Unsupervised learning is where you only have input data (X) and no corresponding output variables . The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data. These are called unsupervised learning because unlike supervised learning above there is no correct answers and there is no teacher. Algorithms are left to their own devises to discover and present the interesting structure in the data.

**Semi-Supervised Learning(Reinforcement Learning)** Problems where you have a large amount of input data (X) and only some of the data is labeled (Y) are called semi-supervised learning problems.

## 3.2 Digital Image Processing

### 3.2.1 Overview

Image processing is a rapidly growing area of computer science. Its growth has been fueled by technological advances in digital imaging, computer pro-

cessors and mass storage devices. Fields which traditionally used analog imaging are now switching to digital systems, for their Fexibility and affordability . Important examples are medicine, Film and video production, photography , remote sensing, and security monitoring. These and other sources produce huge volumes of digital image data every day , more than could ever be examined manually.
Digital image processing is concerned primarily with extracting useful information from images. Ideally , this is done by computers, with little or no human intervention. Image processing algorithms may be placed at three levels. At the lowest level are those techniques which deal directly with the raw, possibly noisy pixel values, with denoising and edge detection being good examples. In the middle are algorithms which utilise low level results for further means, such as segmentation and edge linking. At the highest level are those methods which attempt to extract semantic meaning from the information provided by the lower levels, for example, handwriting recognition.

### 3.2.2 Digital Image

An image may be defined as a two-dimensional function, $f(x, y)$, where x and y are spatial (plane) coordinates, and the amplitude of $f(x, y)$ at any pair of coordinates (x, y) is called the intensity or gray level of the image at that point. When x, y, and the amplitude values of f are all finite, discrete quantities, we call the image a digital image. The field of digital image processing refers to processing digital images by means of a digital computer. Note that a digital image is composed of a finite number of elements, each of which has a particular location and value. These elements are referred to as picture elements, image elements and pixels. Pixel is the term most widely used to denote the elements of a digital image.
A greyscale image measures light intensity only . Each pixel is a scalar proportional to the brightness. The minimum brightness is called black, and the maximum brightness is called white. A typical example is given in fig 3.6
A colour image measures the intensity and chrominance of light. Each color pixel is a vector of colour components. Common color spaces are RGB (red, green and blue), HSV (hue, saturation, value), and CMYK (cyan,magenta, yellow, black), which is used in the printing industry [GW92]. Pixels in a range image measure the depth of distance to an object in the scene. Range data is commonly used in machine vision applications.

Figure 3.6: typical greyscale image

### 3.2.3 Image Acquisition

Before any video or image processing can commence an image must be captured by a camera and converted into a manageable entity. This is the process known as image acquisition. The image acquisition process consists of three steps energy reflected from the object of interest, an optical system which focuses the energy and finally a sensor which measures the amount of energy.

#### Energy

In order to capture an image a camera requires some sort of measurable energy. The energy of interest in this context is light or more generally electromagnetic waves. An electromagnetic (EM) wave can be described as massless entity call photon, the photon belongs to the group of fundamental particles and can be described in three different ways:

- A photon can be described by its energy E, which is measured in electronvolts [EV].
- A photon can be described by its frequency f, which is measured in Hertz [Hz].
- A photon can be described by its wavelength , which is measured in meters [m].

An EM wave can have different wavelengths (or different energy levels or different frequencies). When we talk about all possible wavelengths we denote this as the EM spectrum . the EM spectrum is often described using the names of the applications where they are used in practice .

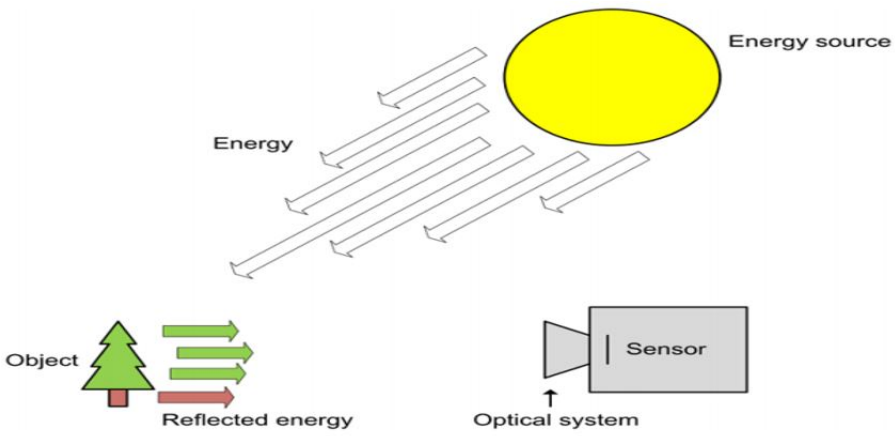



Figure 3.7: typical image acquisition process, with the sun as light source, a tree as object and a digital camera to capture the image. An analog camera would use a film where the digital camera uses a sensor

### Illumination

To capture an image we need some kind of energy source to illuminate the scene in fig 3.7 the sun acts as the energy source. Most often we apply visual light, but other frequencies can also be applied, For example, when tracking the movements of human body parts we use infrared illumination.

### The Optical System

After having illuminated the object of interest, the light reflected from the object now has to be captured by the camera. If a material sensitive to the reflected light is placed close to the object, an image of the object will be captured. However, as illustrated in Fig 3.8 light from different points on the object will mix resulting in a useless image. To make matters worse, light from the surroundings will also be captured resulting in even worse results. The solution is, as illustrated in the figure, to place some kind of

barrier between the object of interest and the sensing material. Note that the consequence is that the image is upside-down.
The hardware and software used to capture the image normally rearranges the image so that you never notice this. The concept of a barrier is a sound idea, but results in too little light entering the sensor. To handle this situation the hole is replaced by an optical system.

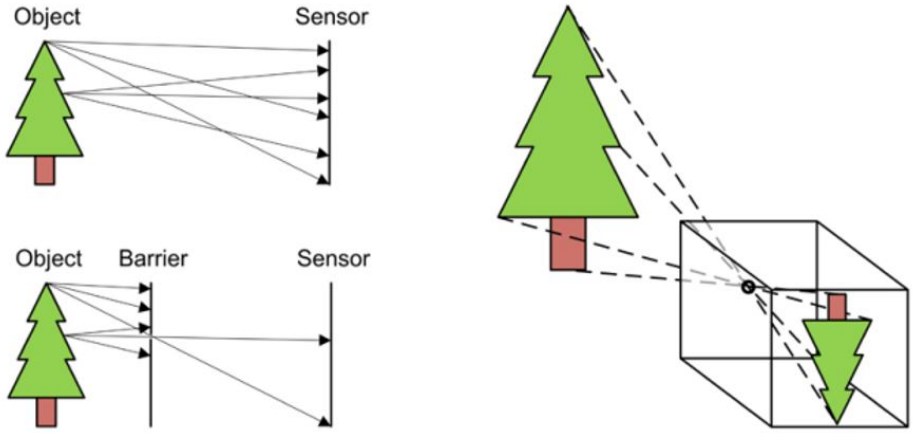



Figure 3.8: Before introducing a barrier, the rays of light from different points on the tree hit multiple points on the sensor and in some cases even the same points. Introducing a barrier with a small hole significantly reduces these problems

### The Image Sensor

The light reflected from the object of interest is focused by some optics and now needs to be recorded by the camera. For this purpose an image sensor is used. An image sensor consists of a 2D array of cells as seen in Fig. 3.9 Each of these cells is denoted a pixel and is capable of measuring the amount of incident light and convert that into a voltage, which in turn is converted into a digital number.
The more incident light the higher the voltage and the higher the digital number. Before a camera can capture an image, all cells are emptied, meaning that no charge is present.
When the camera is to capture an image, light is allowed to enter and charges start accumulating in each cell. After a certain amount of time, known as

the exposure time, and controlled by the shutter, the incident light is shut out again.
The accumulated charges are converted into digital form using an analog to digital converter. This process takes the continuous world outside the camera and converts it into a digital representation, which is required when stored in the computer.

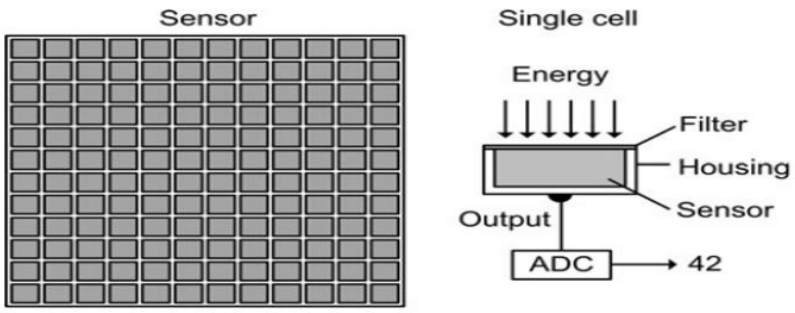



Figure 3.9: The sensor consists of an array of interconnected cells. Each cell consists of a housing which holds a filter, a sensor and an output. The filter controls which type of energy is allowed to enter the sensor. The sensor measures the amount of energy as a voltage, which is converted into a digital number through an analog-to-digital converter (ADC).

### 3.2.4 Image statistics

**The histogram:** An image histogram is a graph of pixel intensity (on the x-axis) versus number of pixels (on the y-axis). The x-axis has all available gray levels, and the y-axis indicates the number of pixels that have a particular gray-level value.

#### Histogram of a Coloured (RGB) Image

The histogram of an RGB image can be displayed in terms of three separate histogramsone for each color component (R, G, and B) of the image. An example is shown in Figure 4. The same information can be represented also by using a 3-D histogram whose axes correspond to the red, green, and blue intensities.

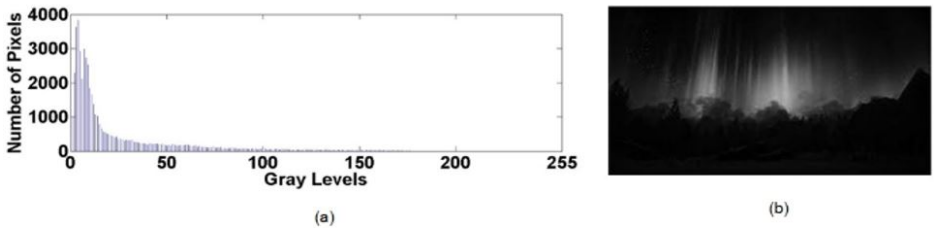



Figure 3.10: histogram of a monochrome image

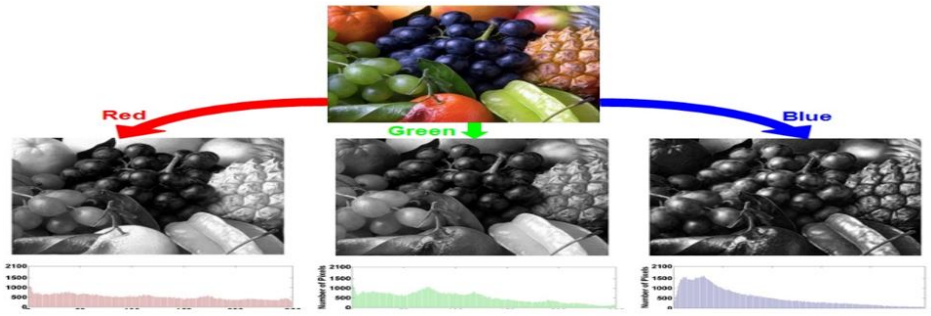



Figure 3.11: Colour image and the histograms corresponding to its red, green and blue monochrome channels

### 3.2.5 Color Space Models

#### The RGB color spaces

The RGB space is a Cartesian coordinate system, the colors are defined as vectors extending from the origin The color of a pixel p is a linear combination of the basis vectors Red, Green and Blue (RGB)where the scalars r, g and b are the RGB components, respectively. The orientation and magnitude of the vector define the chromaticity and intensity of the color, respectivel

### 3.2.6 Image Segmentation and Artificial Neural Network

Segmentation involves partitioning an image into groups of pixels which are homogeneous with respect to some predicate. Each group is called a segment. A good segmentation may be recognized from the characteristics of its output components, each component should be spatially cohesive

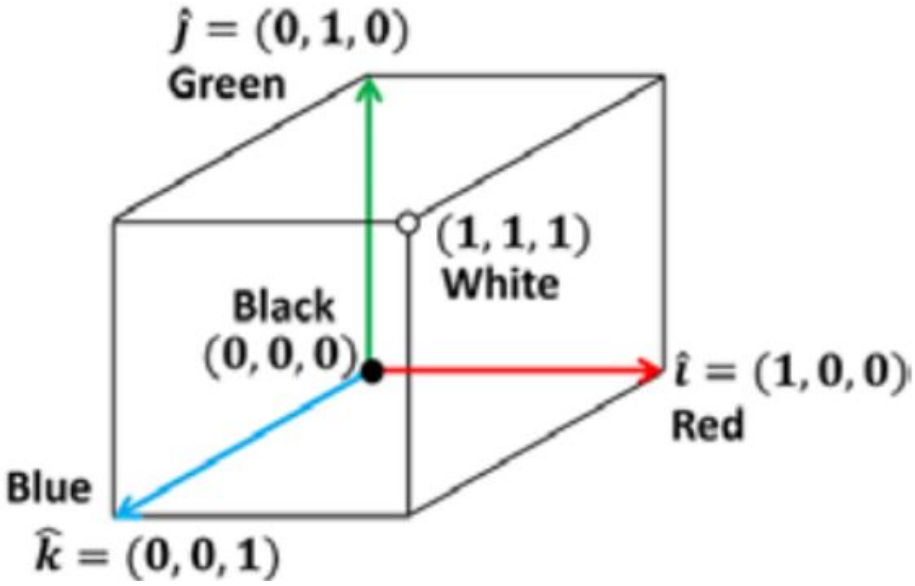



Figure 3.12: RGB Color space.

as well as spatially accurate, while different components should be dissimilar A formal definition of image segmentation is: if $P()$ is a homogeneity defined on groups of connected pixels, then segmentation is a partition of the set F into connected subsets or regions$(S_1, S_2, ., S_n)$ such that: $\cup_{i=1}^{n} S_i = F \quad , \quad \text{with} \quad S_i \cap S_j = \Phi \quad (i \neq j)$
The uniformity predicate $P(S_i) = true$ for all regions , $S_i$ ,and $P(S_i \cup S_j) = false$, when $I \neq j$ and $S_i$ and $S_j$ are neighbors.

Color image segmentation attracts more and more attention . because color images can provide more information than gray level images and the power of personal computers is increasing rapidly.

Since 1990, artificial neural networks have come to be used as a different approach for image segmentation , Almost all types of neural networks have been applied with a different degree of success. The mostly used being Kohonen and Hopfield ANNs. The NN-based image segmentation techniques can mainly be divided into two categories: supervised and unsupervised methods.

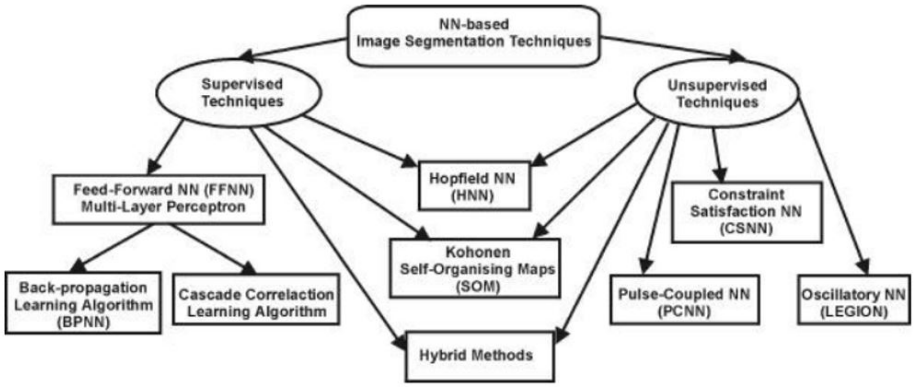



Figure 3.13: ANN-based image segmentation techniques

**3.2.6.1 Supervised techniques** Supervised segmentation techniques fig 3.14 are based on human or operator knowledge to select training images and manually segment them into $k$ regions. Each region is assign with a label and the proposed architecture is trained using the selected images as training data. The method is then able to segment similar images. Labels are assigned to the regions according to the knowledge stored in the NN architecture used.

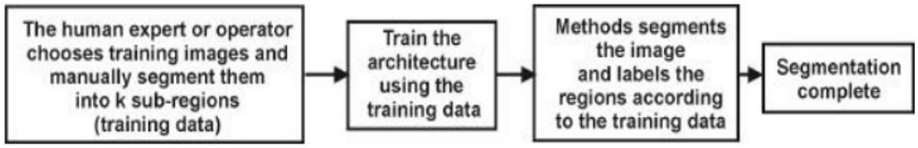



Figure 3.14: supervised ann-based image segmentation technique

### Supervised neural network methods

### Multi-Layer Perceptron (MLP)

Multi-Layer Perceptron (MLP) The simplest NN architecture used for image segmentation is the multi - layer perceptron (MLP). Blanz and Gish[14] had successfully applied a three- layered perceptron to segment grey - level images. Their BPNN approach transformed the image segmentation problem into a pixel classification problem. The input vector for the ANN consisted of

the vector of features extracted from every pixel, and the output vector was the vector of classes desired for segmentation. A standard back- propagation (BP) learning algorithm was chosen to train the network .

Lawson and Parker [14] detailed a very similar approach to image segmentation of industrial radiographic images using back - propagation algorithm. Waard [14] described a MLP trained to classi fy pixels in a text segmentation application. Another back - propagation neural network (BPNN) was implemented by Silverman and Noetzel for detection of tumours in ultrasound medical images.

Coppini et al. [14] described a system based on integrating a priori-anatomical knowledge and a feed- forward BPNN for the segmentation of target structures in tomographic images and of lung nodules in standard projection radiography. Their approach was inspired by the anatomical world: three major blocks were used to segment each type of structure. The input was the retina. Then, the structures were localised by the Attenti on Focuser (AF) and its findings were reported to the output of the system, called the Region Finder (RF). All three blocks were implemented as BPNN, with different parameters. The same approach is described by Valli et al. [14].Even though their perform ance was around 95 correct pixel classification, the presence of noise in the input images was not investigated at all. Their conclusion was that a priorianatomical knowledge could help improve the neural - network based segmentation of medical images.

All supervised methods described so far have a major shortcoming: they all require a training phase using a large set of sample images before run - time. In most of the practical real - time applications, gathering of sample images is an expensive, laborious and time consuming process, if not an almost impossible proces

**3.2.6.2 Unsupervised techniques** An unsupervised segmentation method automatically partitions the images into k sub-regions and then automatically assigns labels to those regions. without operator intervention. A commonly employed method Hopfield NN (HNN), Kohonen Self Organising Maps (SOM), Constraint Satisfaction NN (CSNN), oscillatory NN (LEGION), pulse-coupled NN(PCNN).

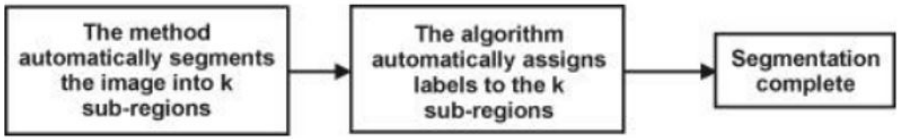



Figure 3.15: Unsupervised ANN-based image segmentation techniques

### Unsupervised neural network methods

### Hopfield ANN (HNN)

A Hopfield network is a form of recurrent artificial neural network popularized by John Hopfield [14] in 1982, but described earlier by Little in 1974. Hopfield nets serve as content-addressable associative memory systems with binary threshold nodes. They are guaranteed to converge to a local minimum, but will sometimes converge to a false pattern (wrong local minimum) rather than the stored pattern (expected local minimum). Hopfield networks also provide a model for understanding human memory.

### Constraint Satisfaction Neural Networks (CSNN)

Lin et al. proposes a new class of neural network for image segmentation Constraint Satisfaction Neural Networks (CSNN). This method is based on the assumption that an image segmentation problem can be described as a Constrained Satisfaction Problem (CSP). The seg mentation problem is redefined as the process of assigning each pixel a label according to certain spatial constraints. A CSNN consists of n x n x mneurons, where n x nis the image size and m is the number of classes the objects needs to be segmented int o (called labels. Each neuron is connected to all its neighbours (8 connections). Those connections are the spatial constraints on the segment label of each pixel. Thus, a CSNN comprises a set of objects, a set of labels (classes) and spatial constraintsdescribing the relationship between various objects according to neighbour relations. They are updated so that a neuron will excite other neurons belonging to the same class and inhibit the others. A winner- take- all scheme is chosen in order to deal with the crossover between segments. As in the standard learning algorithms, there are two phases: learning phase and categorization phase. Each neuron receives feedback from its own output and excitatory/inhibitory signals from its neighbours. This method requires that the number of classes (labels) be

known a priori.

**Pulse-coupled networks or pulse-coupled neural networks (PCNNs)**

are neural models proposed by modeling a cats visual cortex, and developed for high-performance biomimetic image processing.

In 1989, Eckhorn [14] introduced a neural model to emulate the mechanism of cats visual cortex. The Eckhorn model provided a simple and effective tool for studying small mammals visual cortex, and was soon recognized as having significant application potential in image processing.

In 1994, Johnson adapted the Eckhorn model to an image processing algorithm, calling this algorithm a pulse-coupled neural network. Over the past decade, PCNNs have been used in a variety of image processing applications, including: image segmentation, feature generation, face extraction, motion detection, region growing, and noise reduction.

The basic property of the Eckhorn's linking-field model (LFM) is the coupling term. LFM is a modulation of the primary input by a biased offset factor driven by the linking input. These drive a threshold variable that decays from an initial high value. When the threshold drops below zero it is reset to a high value and the process starts over. This is different than the standard integrate-and-fire neural model, which accumulates the input until it passes an upper limit and effectively shorts out to cause the pulse.

LFM uses this difference to sustain pulse bursts, something the standard model does not do on a single neuron level. It is valuable to understand, however, that a detailed analysis of the standard model must include a shunting term, due to the floating voltages level in the dendritic compartment(s), and in turn this causes an elegant multiple modulation effect that enables a true higher-order network (HON). Multidimensional pulse image processing of chemical structure data using PCNN has been discussed by Kinser, et al.

A PCNN is a two-dimensional neural network. Each neuron in the network corresponds to one pixel in an input image, receiving its corresponding pixels color information (e.g. intensity) as an external stimulus. Each neuron also connects with its neighboring neurons, receiving local stimuli from them. The external and local stimuli are combined in an internal activation system, which accumulates the stimuli until it exceeds a dynamic threshold, resulting in a pulse output. Through iterative computation, PCNN neurons produce temporal series of pulse outputs. The temporal series of pulse outputs contain information of input images and can be used for various image

processing applications, such as image segmentation and feature generation. Compared with conventional image processing means, PCNNs have several significant merits, including robustness against noise, independence of geometric variations in input patterns, capability of bridging minor intensity variations in input patterns, etc.

## 3.3 Kohonen Self Organising Maps (SOM)

### 3.3.1 Introduction

In the most artificial neural network topology , the geometrical arrangement of the output is ignored Each neuron in a given layer has identical behavior in that each one receives the signals from the input layer and reacts individually in a similar way. The behavior of one neuron is independent of the behaviors of other neurons in its neighborhood within the layer. In the brain the physical arrangement of neurons is be taken in consideration ,so Nodes that are close together are going to interact differently than nodes that are far apart. neurons tend to cluster in groups. The interactions of the neurons within the group are much greater than those outside the group. The Kohonen self-organizing maps are neural networks that try to mimic this feature in a simple way.
Kohonen self-organizing maps (SOM) are also known as the topology preserving maps, since a topological structure of the output neurons are assumed, and this structure is maintained during the training process. Each output neuron is referred to as a cluster unit. Typically two types of topologies are considered for the output layer , linear and two-dimensional arrays.

Figure 3.16: linear output layer.

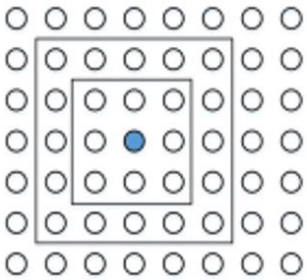


Figure 3.17: two-dimensional array of output neuron arranged on a square lattice.

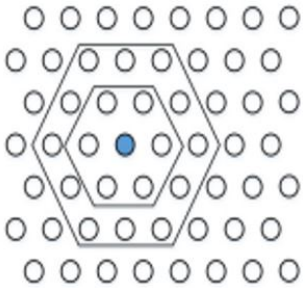


Figure 3.18: two-dimensional array of output neuron arranged on a hexagonal lattice.

### 3.3.2 Self-Organized Systems

Self-organizing systems are types of systems that can change their internal structure and function in response to external circumstances and stimuli . Elements of such a system can influence or organize other elements within the same system, resulting in a greater stability of structure or function of the whole against external fluctuations, . The main aspects of self-organizing systems are increase of complexity, emergence of new phenomena (the whole is more than the sum of its parts) and internal regulation by positive and negative feedback loops. In 1952 Turing published a paper regarding the mathematical theory of pattern formation in biology, and found that global order in a system can arise from local interactions,. This often produces a system with new, emergent properties that differ qualitatively from those of components without interactions . Self-organizing systems exist in nature, including non-living as well as living world, they exist in man-made systems, but also in the world of abstract ideas[10].

### 3.3.3 Self Organising Maps

The Self-Organizing Maps (SOM) algorithm, also called the Kohonen network, the self organizing feature map(SOFM) or the topological map, is an unsupervised neural network algorithm developed by the Finish physicist Teuvo Kohonen (1980). It has biological background on nervous systems .

It provides two useful operations in exploratory data analysis: clustering, reducing the amount of data into representative categories, and non-linear projection, aiding in the exploration of proximity relations in the data. The topology preservation is accomplished through an ordered mapping in such a way that nearby patterns in the input space are mapped also to nearby units in output space. In this respect, the SOM is a multidimensional scaling method, which projects data from input space to a lower dimensional output space.
The SOM consists of an input layer having N neurons, and an output layer having M neurons (cluster units) arranged in some predetermined fashion. Each neuron in the input layer is connected to a neuron in the output layer figure. Thus output neuron usually arranged in a low dimensionality grid (1-D or 2-D), the map, for ease of visualization. The lattice(map or grid) type may have several forms like rectangular, hexagonal or even irregular.

Consider an input dataset R containing n instances,$R = \{X_1, X_2, X_3, , X_s\}$ , Each instance Xi is represented by a n-dimensional attribute vector (instance vector), $X_i = \{X_{i,1}, X_{i,2}, ., X_{i,n}\}$. Let k be the number of neurons in the feature map$M, M = m_1, m_2, m_3, , m_k$. Each neuron is associated with a n-dimensional weight vector (neuron vector) having the same dimensionality as the input instance vector. Initially the neuron weight vectors are initialized randomly. All input instances are presented to the SOM one by one. A complete pass over the input set is called an epoch,$0 \leq t \leq n$. A complete training of the SOM requires multiple passes over the input data (multiple epochs).

The training utilizes competitive learning. When a training example is fed to the network, its Euclidean distance to all weight vectors is computed. The neuron whose weight vector is most similar to the input is called the best matching unit (BMU). The weights of the BMU and neurons close to it in the SOM grid are adjusted towards the input vector. After identifying the winner neuron, all neuron vectors in the map are adapted to the presented input $X_i$. The degree of adaptation with respect to the input instance as well as the distance from the winner and the learning rate.

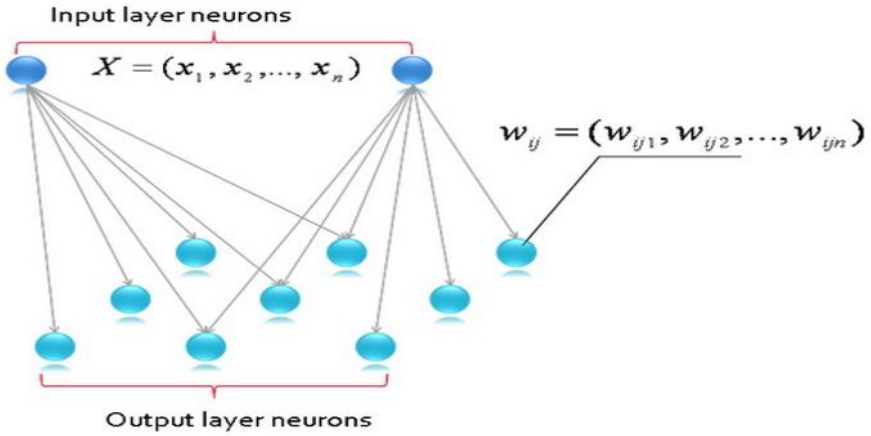



Figure 3.19: The architecture of an SOFM NN

**Measures of Distance and Similarity**

To determined similarity between the input vector and neurons measures of distance are used. Some popular distances among input pattern and SOM units are:

- Euclidian
- Correlation
- Direction cosine
- Block distance

In a real application most often squared Euclidean distance is used :

$$d_j = \sum_i (x_i - w_{ij})^2$$

### 3.3.4 Training of self-organizing maps

Training of self-organizing maps can be accomplished in two ways: as Serial or Batch training

**Sequential training (Serial or On-Line)**

We assume a set of input vectors $x \in R_n$in a n-dimensional space, and a weight vector $W \in R_n$ for each neuron $k$ in a regular 2D grid of $K$ neurons see fig 3.20. A discrete temporal index t exists, such that a vector $x(t)$ is presented to the network in time t, and $W_k(t)$ is the weight vector calculated for that instant. Input patterns are recycled during training; one single pass over the training set is called an epoch. The initial values for the weight vectors can be set randomly or using K different input patterns. In this method the reference vectors are updated immediately upon the presentation of an input pattern.
The two main stages of the SOM training are :

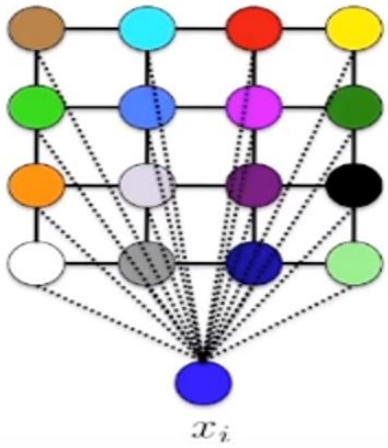



Figure 3.20: every node is connected to the input the same way

1. **Finding the best matching neuron (winner)** At a given time instant t, a random input instance $x(t)$ is presented to the neuron map. $W_k(t)$ is the weight vector of the neuron $W_k$ at instant t. The input instance is compared with each neuron using a distance metric d (usually the Euclidean distance for a Euclidean feature space).

$$d_j = \sum_i (x_i - w_{ij})^2 \tag{3.1}$$

The neuron with the minimum distance from the instance is selected as the winner neuron $c(t)$.

$$c(t) = \arg min_k d_k(t) \tag{3.2}$$

2. **Adaptation of weight vectors** After identifying the winner neuron, all neuron vectors in the map are adapted to the After identifying the winner neuron, all neuron vectors in the map are adapted to the presented input $x(t)$. The degree of adaptation with respect to the input instance gradually decreases with t as well as the distance from the winner,$c(t)$. In othe words, at $t = 0$, almost all the neurons in the neuron map are updated, while towards the end of an epoch, only the winner neuron is updated. The neuron vectors in the map are updated. (adapted to the input instance $x(t)$ using,

$$W_k(t+1) = W_k(t) + \alpha(t)h_{ck}(t)[x(t) - W_k(t)] \tag{3.3}$$

$\alpha(t)$ is the learning rate , The learning rate controls the amount of correction to the neuron vectors and decreases with time .

$$\alpha(t) = \alpha_0 e^{-\frac{t}{T}} \tag{3.4}$$

**The neighbourhood function** controls the number of units that are adapted as well as the degree of adaptation. During an epoch, the neurons closer to the winner are adapted more to x(t) than the neurons further from the winner. Usually the standard Gaussian neighbourhood function is used for computing the neighbourhood factor.

$$h_{ck}(t) = e^{\frac{d^2}{2r^2}} \tag{3.5}$$

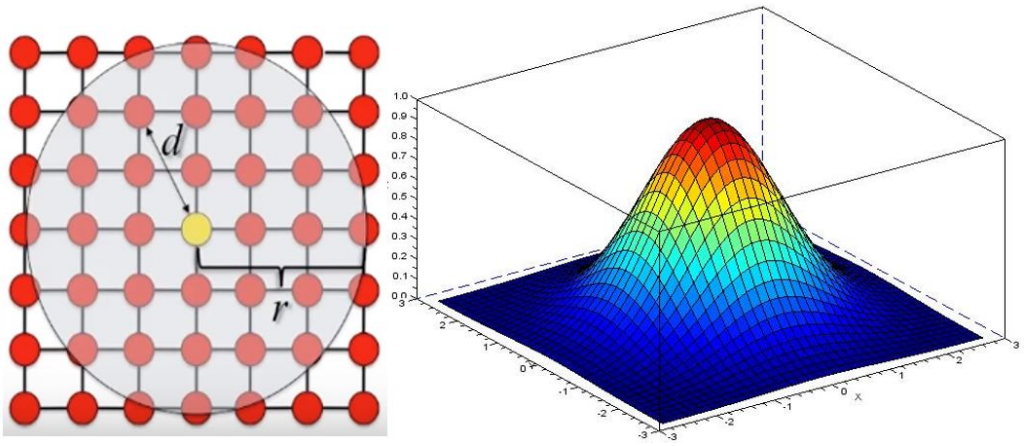



Figure 3.21: The standard Gaussian neighbourhood function

$r$ time the radius $r$ in equations 3.5 will shrink to the size by $r(t)$ from the BMU.

$$r(t) = (MAXSIZE)e^{-t\frac{\ln(MAXSIZE)}{T}} \tag{3.6}$$

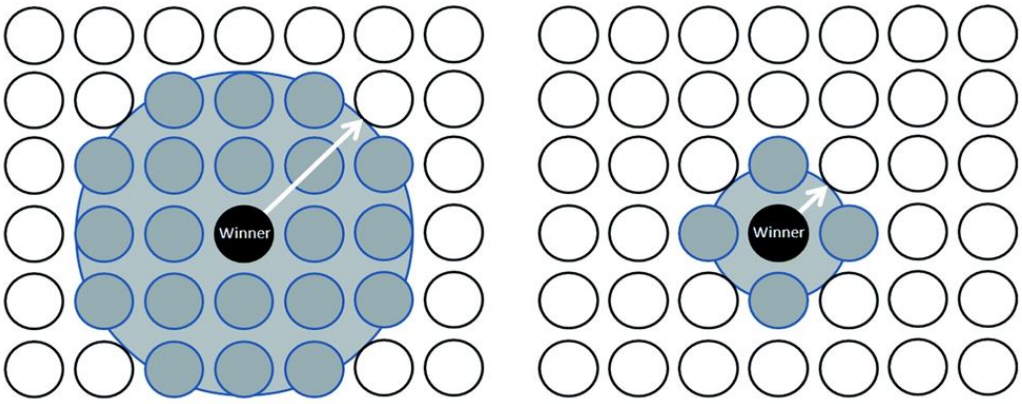



Figure 3.22: Shrinking radius. left map R(t), right map r(t+n).

To summarize, for every input instance, the winner neuron is identified and the weight vectors of all neurons are adapted with respect to this instance. The Figure 3.23 outlines the adjustment phase of

the SOM. The black neuron being the winner is adapted the most.The amount of adaptation to the input is depicted by the color gradient (the black colour indicating the most impact while the white colour indicating the least impact).
This variant of the SOM described above is called the Serial SOM or the Online SOM. It adjusts the neurons on every presentation of an input. The Serial SOM Algorithm can be formalized as follows:

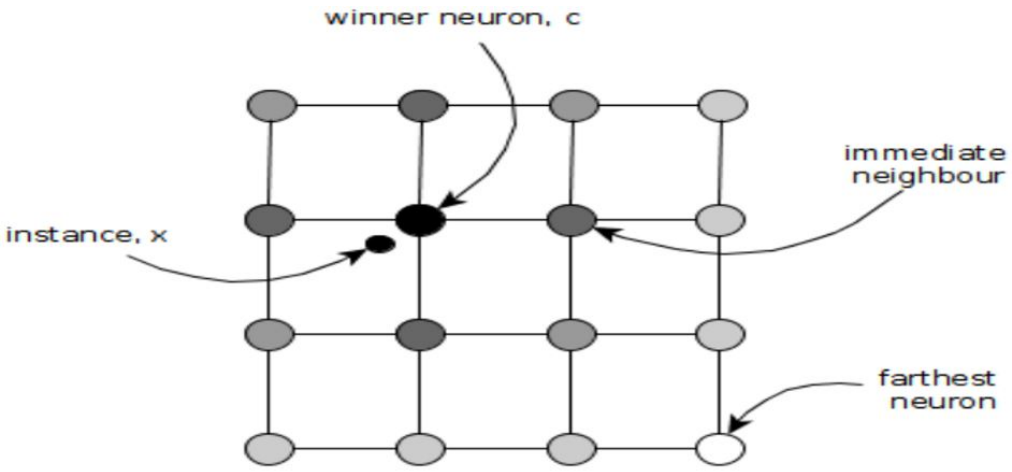



Figure 3.23: Adaptation of neurons to input instance.

(a) Find the BMU for a input vector x(t), using equations 3.1 and 3.2

(b) Update the weight vectors of the neurons with equations 3.3 and 3.4

(c) Repeat from step 1 until some convergence criterion is met, with decreasing neighborhood kernel using equations 3.5 .

**Batch training (Off-Line)**

in the Batch SOM, we do not need to provide the learning rate (t), which is susceptible to poor convergence when not selected properly . Moreover, the Batch SOM algorithm is well-suited for parallelization. Batches of input data can be processed in parallel and neuron vectors can be updated at the

end of all batches. and the new weights are computed using:

$$W_i(t_f) = \frac{\sum_{t'=t_0}^{t'=t_f} h_{ck}(t')x(t')}{\sum_{t'=t_0}^{t'=t_f} h_{ck}(t')} \tag{3.7}$$

here $t_0$ and $t_f$ stand for, respectively, for the beginning and end of the current batch. The so-called winner is found with equations 3.1 and 3.2 .
$W_k(t_0)$ is the neuron weight vector calculated at the end of the previous epoch. The neighborhood $h_{ci}$ function controls the number of units that are adapted as well as the degree of adaptation. To calculate this neighborhood region, we use the exponential Gaussian neighborhood function which is a decreasing function of time. Hence, the radius of the neighborhood gradually decreases as the algorithm progresses.

$$h_{ck}(t) = e^{\frac{d^2}{2r^2}} \tag{3.8}$$

$$r(t) = r_0(1 - \frac{t}{T})$$

**The Batch SOM algorithm can be stated as follows:**

1. For each epoch:
   (a) Find the BMU for an input vector x(t), using 3.1 and 3.2
   (b) Accumulate numerator and denominator of equation 3.7 for all neurons
2. Update neuron weights with equation 3.7
3. Repeat from step 1 until some convergence criterion is met, with decreasing neighborhood kernel.

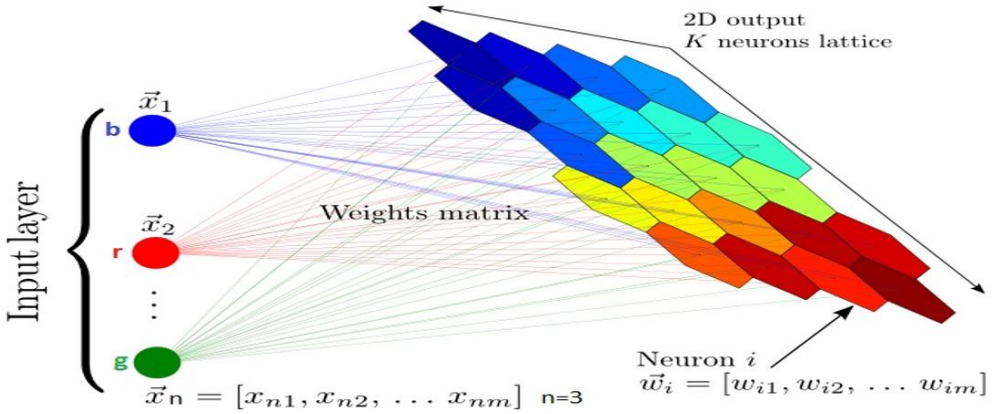
The Sequential algorithm is commonly used with the Network-Partition approach. With it, each processing node as to deal with every input pattern, using only the part of the map that as been assign to it. Since the maps are smaller, the local search is done much faster. The main advantage of this approach is that it preserves the original weight update method , thus being able to reproduce exactly the results of the Sequential SOM algorithm. However it makes it necessary that all nodes communicate with each other at

every iteration when needing the BMU for the current input pattern. This frequent communication limits scalability and introduces a constant latency in the processing of each input. Several implementations have been devised using this approach in different architectures .

Using the Data-Partition approach overs no clear advantages, because the processing nodes still need to communicate at every iteration, not to elect the BMU for an input pattern, but to update the state of the map. Furthermore, since the resulting map is dependent of the order in which the input patterns are presented to the network, this approach would have different results with different data segmentations. As for the Batch algorithm the data-partitioning approach is particularly well suited, allowing greater scalability, since parallel granularity is determined by the volume of data, which is potentially large. The method partitions the set of input patterns equally along the nodes, and each node trains a complete copy of the map, needing only to communicate at the end of each epoch to update the map . The Batch Data-Partitioning method overs a considerable speed-up in the learning process, against the Sequential Network-Partitioning, due to less inter-node communication. Some authors point out that the quality of the clustering produced by the Batch algorithm is inferior to the one of the Sequential algorithm , but using a k-Batch modification the results can be very similar, forcing the updates do be made k times during each batch . the information that can be extracted from the Batch map is exactly the same. We consider that these minor differences are acceptable, given the computing time that can be saved in the training process, making it the best solution for time-critical analysis.

### Application of SOM to Color Image Segmentation

**SOM Training** For each pixel V of the image, the Best Matching Unit (BMU) is selected in the SOM. BMU has the least dissimilarity with the input pixel from the rest of the SOM. The weight associated with this neuron is updated according to 3. New weight Wi of the neuron i at iteration t+1 is calculated. Here $\alpha_0$ is the initial learning rate, $r_0$ is the initial neighborhood size . For each neuron within the neighborhood of i,$h_{ci}(t)$ is calculated using **??** . Then the weight vector of all neurons within the neighborhood r(t), at time $t$, is updated using 3.7. Neurons nearer to the BMU becomes more alike the BMU then those neurons farther but within the neighborhood. After all the pixels of the image are presented to the SOM then t is incremented by 1. This process is repeated unless t reaches tmax. SOM after training on the river image look like the figure bellow.




**Segmentation (classification of pixels)** After training the randomly initialized Serial SOM . For each pixel in the image, a BMU is found in the SOM. Then the value of the image pixel is replaced by the BMU from SOM. This results in the final segmented image.

## 3.4 Parallel computing

### 3.4.1 introduction

arallel computing is a type of computation in which many calculations or the execution of processes are carried out concurrently. Large problems can often be divided into smaller ones, which can then be solved at the same time. There are several different forms of parallel computing: bit-level, instruction-level, data, and task parallelism. Parallelism has long been employed in high-performance computing, but it's gaining broader interest due to the physical constraints preventing frequency scaling. As power consumption (and consequently heat generation) by computers has become a concern in recent years, parallel computing has become the dominant paradigm in computer architecture, mainly in the form of multi-core processors.

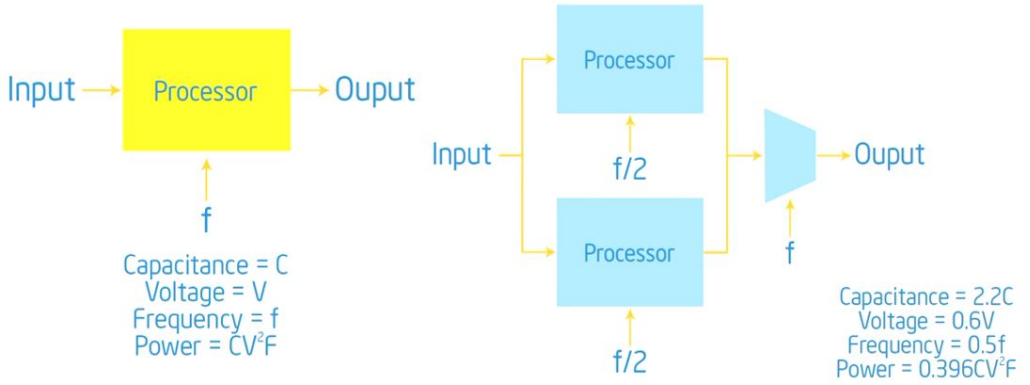



Figure 3.24: comparision between a single-core and a multi-core processor architecture , the multi-core design is 39% more power efficient.

Parallel computing is closely related to concurrent computing they are frequently used together, and often conflated, though the two are distinct: it is possible to have parallelism without concurrency (such as bit-level parallelism), and concurrency without parallelism (such as multitasking by time-sharing on a single-core CPU). In parallel computing, a computational task is typically broken down into several, often many, very similar subtasks that can be processed independently and whose results are combined afterwards, upon completion. In contrast, in concurrent computing, the various processes

often do not address related tasks; when they do, as is typical in distributed computing, the separate tasks may have a varied nature and often require some inter-process communication during execution.

Parallel computers can be roughly classified according to the level at which the hardware supports parallelism, with multi-core and multi-processor computers having multiple processing elements within a single machine, while clusters, MPPs, and grids use multiple computers to work on the same task. Specialized parallel computer architectures are sometimes used alongside traditional processors, for accelerating specific tasks.In some cases parallelism is transparent to the programmer, such as in bit-level or instruction-level parallelism, but explicitly parallel algorithms, particularly those that use concurrency, are more difficult to write than sequential ones, because concurrency introduces several new classes of potential software bugs, of which race conditions are the most common. Communication and synchronization between the different subtasks are typically some of the greatest obstacles to getting good parallel program performance.

### 3.4.2 Classification of parallel computers

#### Flynn's taxonomy

This classification was first studied and proposed by Michael Flynn in 1972. Flynn did not consider the machine architecture for classification of parallel computers; he introduced the concept of instruction and data streams for categorizing of computers. All the computers classified by Flynn are not parallel computers, but to grasp the concept of parallel computers, this classification is based on instruction and data streams.
first we need to understand how the instruction cycle works.

**Instruction Cycle** The instruction cycle consists of a sequence of steps needed for the execution of an instruction in a program. A typical instruction in a program is composed of two parts: Opcode and Operand. The Operand part specifies the data on which the specified operation is to be done. The Operand part is divided into two parts: addressing mode and the Operand. The addressing mode specifies the method of determining the addresses of the actual data on which the operation is to be performed and the operand part is used as an argument by the method in determining the actual address.

The control unit of the CPU of the computer fetches instructions in the pro-

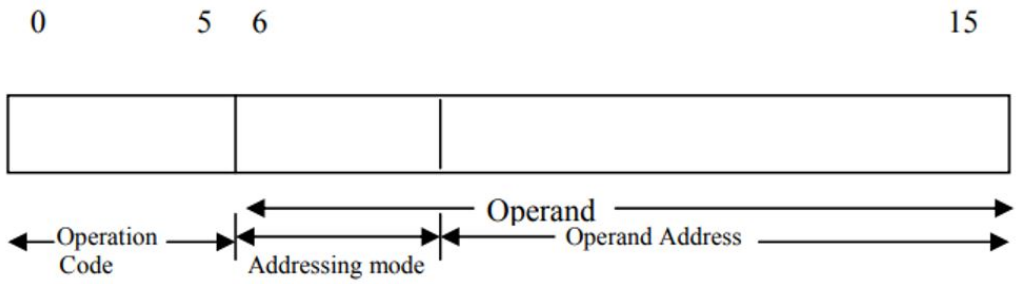



Figure 3.25: Opcode and Operand.

gram, one at a time. The fetched Instruction is then decoded by the decoder which is a part of the control unit and the processor executes the decoded instructions. The result of execution is temporarily stored in Memory Buffer Register. The normal execution steps are shown Figure below .

**Instruction Stream and Data Stream** The term stream refers to a sequence or flow of either instructions or data operated on by the computer. In the complete cycle of instruction execution, a flow of instructions from main memory to the CPU is established. This flow of instructions is called instruction stream. Similarly, there is a flow of operands between processor and memory bi-directionally. This flow of operands is called data stream. These two types of streams are shown in Figure 3.27 . it can be said that the sequence of instructions executed by CPU forms the Instruction streams and sequence of data (operands) required for execution of instructions form the Data streams.

**Flynns Classification**

Flynns classification is based on multiplicity of instruction streams and data streams observed by the CPU during program execution. Let Is and Ds are minimum number of streams flowing at any point in the execution, then the computer organisation can be categorized as follows:

1. **Single Instruction and Single Data stream (SISD)** inthis organisation, sequential execution of instructions is performed by one CPU containing a single processing element , ALU under one control unit .

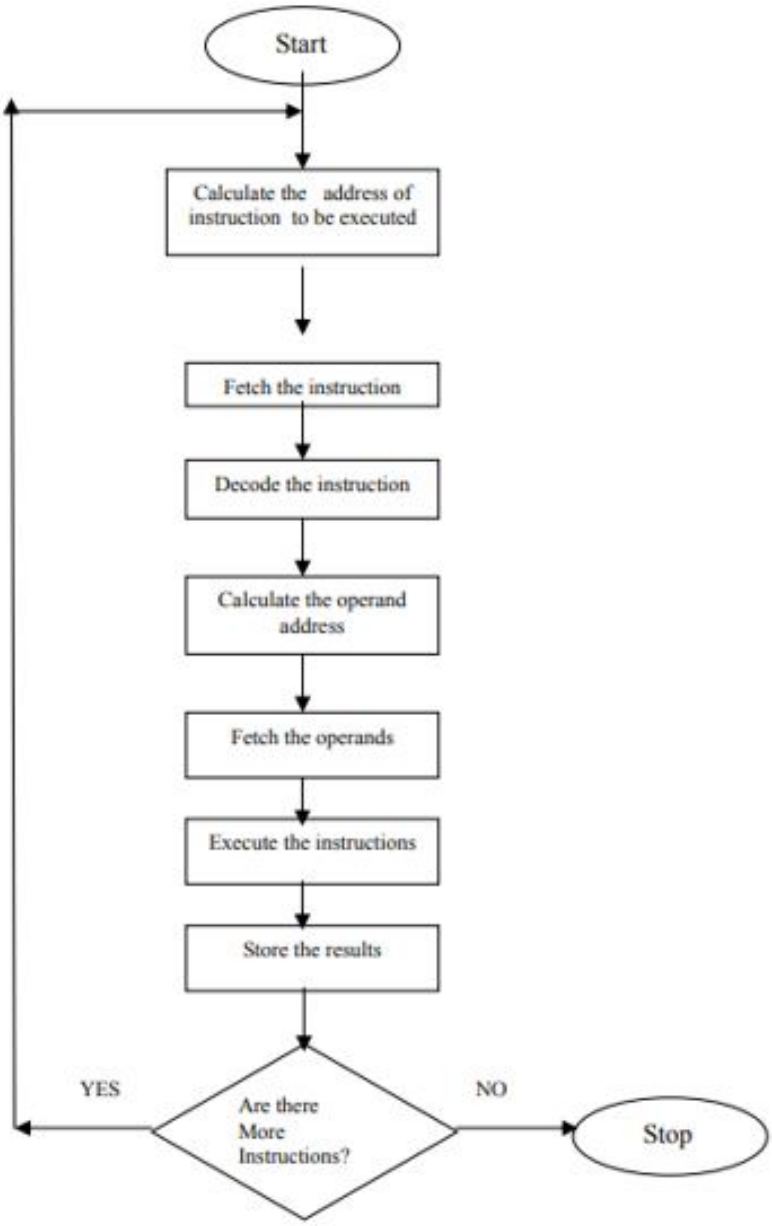



Figure 3.26: Instruction execution steps

Therefore, SISD machines are conventional serial computers that process only one stream of instructions and one stream of data on by the

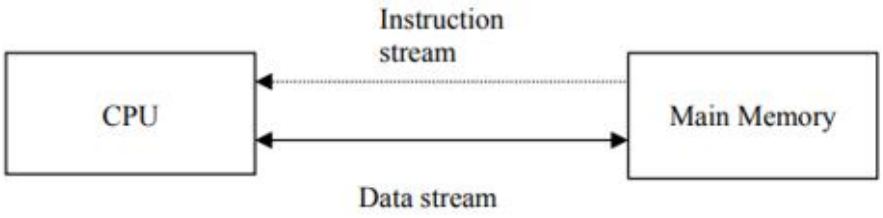



Figure 3.27: Instruction and data stream

CPU during one clock cycle . This type of computer organisation is depicted in the diagram: $I_s = D_s = 1$

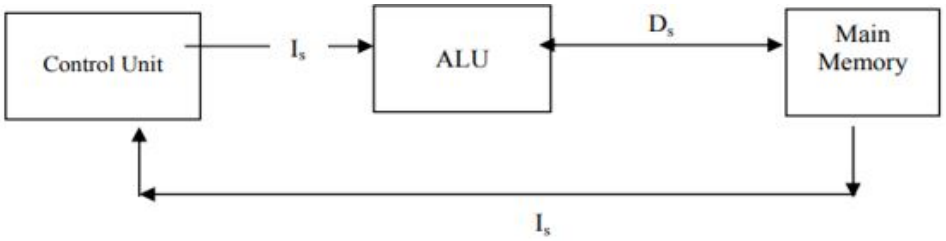



Figure 3.28: SISD organisation

Examples of SISD machines include:


UNIVAC1


IBM 360


CRAY1


CDC 7600


PDP1


Dell Laptop

2. **Single Instruction and Multiple Data stream (SIMD)** In this organisation, multiple processing elements work under the control of a single control unit. It has one instruction and multiple data stream. All the processing elements of this organization receive the same instruction broadcast from the CU. Main memory can also be divided into modules for generating multiple data streams acting as a distributed memory as shown in Figure 5. Therefore, all the processing elements simultaneously execute the same instruction and are said to be 'lock-stepped' together. Each processor takes the data from its own memory and hence it has on distinct data streams. (Some systems also provide a shared global memory for communications.) Every processor must be allowed to complete its instruction before the next instruction is taken for execution. Thus, the execution of instructions is synchronous. Examples of SIMD organisation are ILLIAC-IV, PEPE, BSP, STARAN, MPP, DAP and the Connection Machine (CM-1). This type of computer organisation is denoted as:

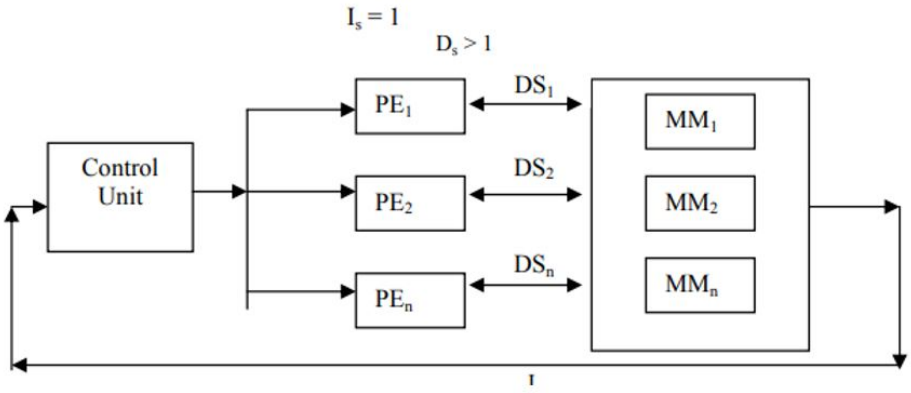
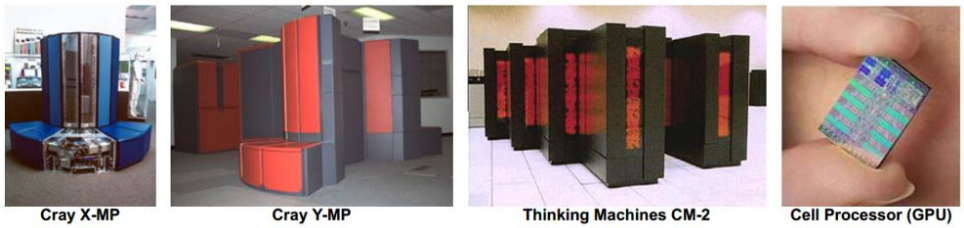



Figure 3.29: SIMD organisation

Examples of SIMD machines include:




3. **Multiple Instruction and Single Data stream (MISD)** In this organization, multiple processing elements are organised under the control of multiple control units. Each control unit is handling one instruction stream and processed through its corresponding processing element. But each processing element is processing only a single data stream at a time. Therefore, for handling multiple instruction streams and single data stream, multiple control units and multiple processing elements are organised in this classification. All processing elements are interacting with the common shared memory for the organisation of single data stream . The only known example of a computer capable of MISD operation is the C.mmp built by Carnegie-Mellon University. This type of computer organisation is denoted as: $I_s > 1 \quad D_s = 1$ This classification is not popular in commercial machines as the concept of single data streams executing on multiple processors is rarely applied.

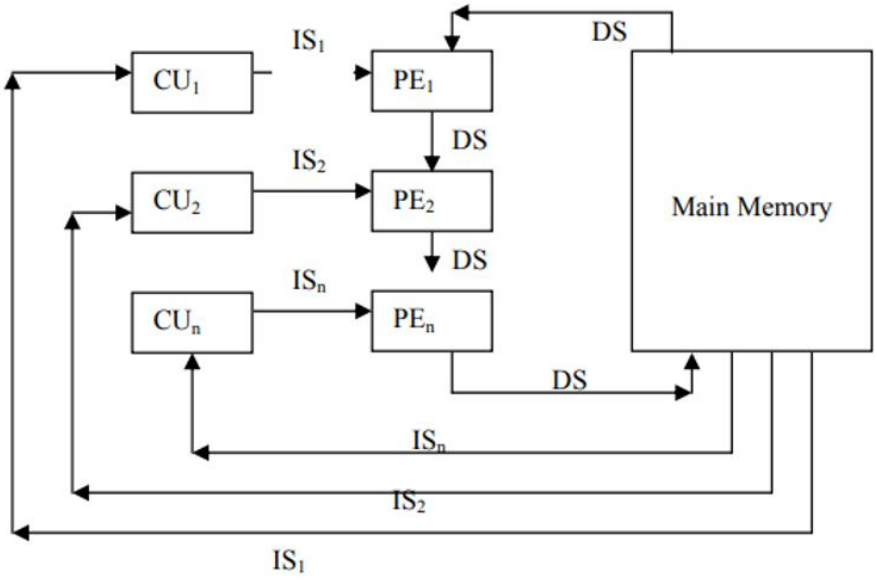



Figure 3.30: MISD organisation

But for the specialized applications, MISD organisation can be very helpful. For example, Real time computers need to be fault tolerant where several processors execute the same data for producing the redundant data. This is also known as N- version programming. All these redundant data are compared as results which should be same; otherwise faulty unit is replaced. Thus MISD machines can be applied to fault tolerant real time computers.

4. **Multiple Instruction and Multiple Data stream (MIMD)** In this organization, multiple processing elements and multiple control units are organized as in MISD. But the difference is that now in this organization multiple instruction streams operate on multiple data streams . Therefore, for handling multiple instruction streams, multiple control units and multiple processing elements are organized such that multiple processing elements are handling multiple data streams from the Main memory . The processors work on their own data with their own instructions. Tasks executed by different processors can start or finish at different times. They are not lock-stepped, as in SIMD computers, but run asynchronously. This classification actually recognizes the parallel computer. That means in the real sense MIMD organisation

is said to be a Parallel computer. All multiprocessor systems fall under this classification. Examples include; C.mmp, Burroughs D825, Cray-2, S1, Cray X-MP, HEP, Pluribus, IBM 370/168 MP, Univac 1100/80, Tandem/16, IBM 3081/3084, C.m*, BBN Butterfly, Meiko Computing Surface (CS-1), FPS T/40000, iPSC.
This type of computer organisation is denoted as:
Of the classifications discussed above, MIMD organization is the most

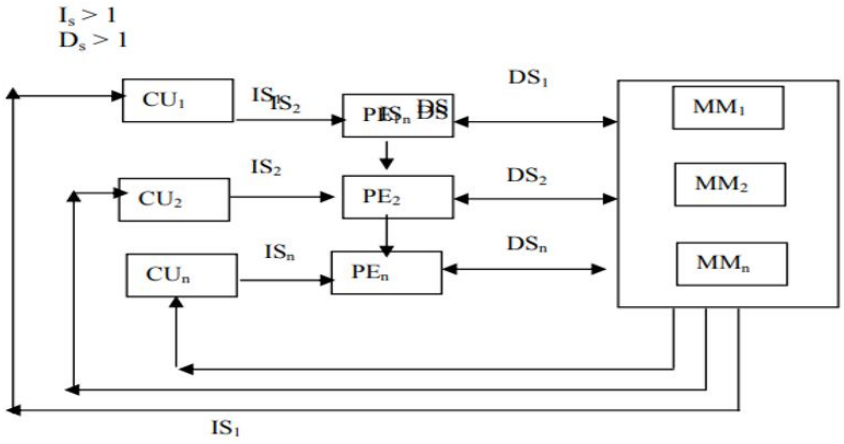



Figure 3.31: MIMD organisation

popular for a parallel computer, In the real sense, parallel computers execute the instructions in MIMD mode.

### 3.4.3 Types of parallelism

#### Bit-level parallelism

From the advent of very-large-scale integration (VLSI) computer-chip fabrication technology in the 1970s until about 1986, speed-up in computer architecture was driven by doubling computer word sizethe amount of information the processor can manipulate per cycle. Increasing the word size reduces the number of instructions the processor must execute to perform an operation on variables whose sizes are greater than the length of the word. For example, where an 8-bit processor must add two 16-bit integers, the processor must first add the 8 lower-order bits from each integer using the

standard addition instruction, then add the 8 higher-order bits using an add-with-carry instruction and the carry bit from the lower order addition; thus, an 8-bit processor requires two instructions to complete a single operation, where a 16-bit processor would be able to complete the operation with a single instruction.
Historically, 4-bit microprocessors were replaced with 8-bit, then 16-bit, then 32-bit microprocessors. This trend generally came to an end with the introduction of 32-bit processors, which has been a standard in general-purpose computing for two decades. Not until the early 2000s, with the advent of x86-64 architectures, did 64-bit processors become commonplace.

### Instruction-level parallelism

A computer program is, in essence, a stream of instructions executed by a processor. Without instruction-level parallelism, a processor can only issue less than one instruction per clock cycle (IPC ¡ 1). These processors are known as subscalar processors. These instructions can be re-ordered and combined into groups which are then executed in parallel without changing the result of the program. This is known as instruction-level parallelism. Advances in instruction-level parallelism dominated computer architecture from the mid-1980s until the mid-1990s.
All modern processors have multi-stage instruction pipelines. Each stage in


Figure 3.32: A canonical processor without pipeline. It takes five clock cycles to complete one instruction and thus the processor can issue subscalar performance (IPC = 0.2 ).

the pipeline corresponds to a different action the processor performs on that instruction in that stage; a processor with an N-stage pipeline can have up to N different instructions at different stages of completion and thus can issue one instruction per clock cycle (IPC = 1). These processors are known as scalar processors. The canonical example of a pipelined processor is a RISC processor, with five stages: instruction fetch (IF), instruction decode

(ID), execute (EX), memory access (MEM), and register write back (WB). The Pentium 4 processor had a 35-stage pipeline. Most modern processors

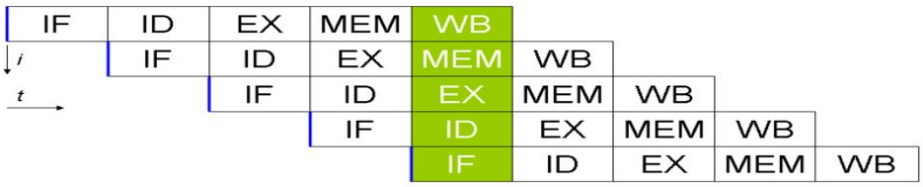



Figure 3.33: a canonical five-stage pipelined processor. in the best case scenario, it takes one clock cycle to complete one instruction and thus the processor can issue scalar performance (ipc = 1).

also have multiple execution units. They usually combine this feature with pipelining and thus can issue more than one instruction per clock cycle (IPC ¿ 1). These processors are known as superscalar processors. Instructions can be grouped together only if there is no data dependency between them. Scoreboarding and the Tomasulo algorithm (which is similar to scoreboarding but makes use of register renaming) are two of the most common techniques for implementing out-of-order execution and instruction-level parallelism.

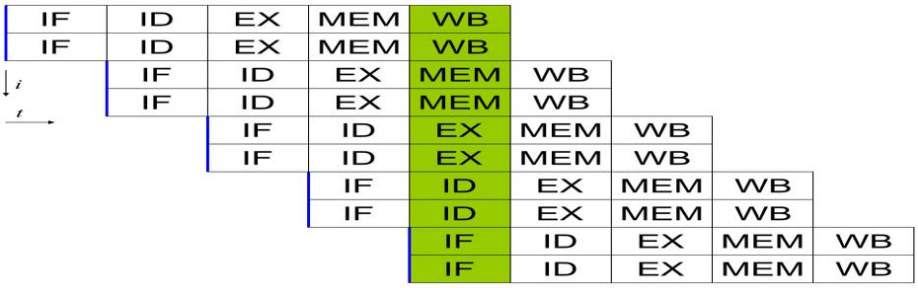



Figure 3.34: A canonical five-stage pipelined superscalar processor. In the best case scenario, it takes one clock cycle to complete two instructions and thus the processor can issue superscalar performance (IPC = 2 ).

### Task parallelism

Task parallelisms is the characteristic of a parallel program that entirely different calculations can be performed on either the same or different sets

of data. This contrasts with data parallelism, where the same calculation is performed on the same or different sets of data. Task parallelism involves the decomposition of a task into sub-tasks and then allocating each sub-task to a processor for execution. The processors would then execute these sub-tasks concurrently and often cooperatively. Task parallelism does not usually scale with the size of a problem.

### 3.4.4 Classes of parallel computers

Parallel computers can be roughly classified according to the level at which the hardware supports parallelism. This classification is broadly analogous to the distance between basic computing nodes.

#### Multi-core computing

A multi-core processor is a processor that includes multiple processing units (called "cores") on the same chip. This processor differs from a superscalar processor, which includes multiple execution units and can issue multiple instructions per clock cycle from one instruction stream (thread); in contrast, a multi-core processor can issue multiple instructions per clock cycle from multiple instruction streams. IBM's Cell microprocessor, designed for use in the Sony PlayStation 3, is a prominent multi-core processor. Each core in a multi-core processor can potentially be superscalar as wellthat is, on every clock cycle, each core can issue multiple instructions from one thread. Concurrent multithreading (of which Intel's Hyper-Threading is the best known) was an early form of pseudo-multi-coreism. A processor capable of concurrent multithreading includes multiple execution units in the same processing unit that is it has a superscalar architecture and can issue multiple instructions per clock cycle from multiple threads. Temporal multithreading on the other hand includes a single execution unit in the same processing unit and can issue one instruction at a time from multiple threads.

#### Distributed Computing

A distributed computer (also known as a distributed memory multiprocessor) is a distributed memory computer system in which the processing elements are connected by a network. Distributed computers are highly scalable. The terms concurrent computing, parallel computing , and distributed computing

have a lot of overlap, and no clear distinction exists between them The same system may be characterized both as parallel and distributed; the processors in a typical distributed system run concurrently in parallel.

### Cluster Computing

A cluster is a group of loosely coupled computers that work together closely, so that in some respects they can be regarded as a single computer. Clusters are composed of multiple standalone machines connected by a network. While machines in a cluster do not have to be symmetric, load balancing is more difficult if they are not. The most common type of cluster is the Beowulf cluster, which is a cluster implemented on multiple identical commercial off-the-shelf computers connected with a TCP/IP Ethernet local area network. Beowulf technology was originally developed by Thomas Sterling and Donald Becker. The vast majority of the TOP500 supercomputers are clusters.
Because grid computing systems (described below) can easily handle embarrassingly parallel problems, modern clusters are typically designed to handle more difficult problemsproblems that require nodes to share intermediate results with each other more often. This requires a high bandwidth and, more importantly, a low-latency interconnection network. Many historic and current supercomputers use customized high-performance network hardware specifically designed for cluster computing, such as the Cray Gemini network. As of 2014, most current supercomputers use some off-the-shelf standard network hardware, often Myrinet, InfiniBand, or Gigabit Ethernet.

### Grid Computing

Grid computing is the most distributed form of parallel computing. It makes use of computers communicating over the Internet to work on a given problem. Because of the low bandwidth and extremely high latency available on the Internet, distributed computing typically deals only with embarrassingly parallel problems. Many distributed computing applications have been created, of which SETI@home and Folding@home are the best-known examples.
Most grid computing applications use middleware (software that sits between the operating system and the application to manage network resources and standardize the software interface). The most common distributed comput-

ing middleware is the Berkeley Open Infrastructure for Network Computing (BOINC). Often, distributed computing software makes use of "spare cycles", performing computations at times when a computer is idling.

### Massively Parallel Computing

A massively parallel processor (MPP) is a single computer with many networked processors. MPPs have many of the same characteristics as clusters, but MPPs have specialized interconnect networks (whereas clusters use commodity hardware for networking). MPPs also tend to be larger than clusters, typically having far more than 100 processors. In an MPP, each CPU contains its own memory and copy of the operating system and application. Each subsystem communicates with the others via a high-speed interconnect.

### General-purpose computing on graphics processing units (GPGPU)

General-purpose computing on graphics processing units (GPGPU) is a fairly recent trend in computer engineering research. GPUs are co-processors that have been heavily optimized for computer graphics processing. Computer graphics processing is a field dominated by data parallel operations particularly linear algebra matrix operations.
In the early days, GPGPU programs used the normal graphics APIs for executing programs. However, several new programming languages and platforms have been built to do general purpose computation on GPUs with both Nvidia and AMD releasing programming environments with CUDA and Stream SDK respectively. Other GPU programming languages include BrookGPU, PeakStream, and RapidMind. Nvidia has also released specific products for computation in their Tesla series. The technology consortium Khronos Group has released the OpenCL specification, which is a framework for writing programs that execute across platforms consisting of CPUs and GPUs. AMD, Apple, Intel, Nvidia and others are supporting OpenCL.
4.4.6 Parallel Programming and its Challenges on the Multicore Processors
The parallelization of the program or software for the multicore processors is a very difficult task for programmers and algorithm developers. Some of the task is not possible to convert into parallel completely. This is not possible due to the lack of non-availability of the algorithm or subject expertise. Otherwise, the output of the program may be changed and incorrect. The parallelization of program may increase the space complexity, time complex-

ity. But it may improve the throughput time of the program. So, the output of the parallel program can be obtained in a less time in comparison of the sequential program.
The most of the algorithms have been implemented with sequential programming, since in the past the parallel programming concept was not available to the developers. For converting the sequential program into parallel program, the algorithms have to be converted into parallel and implement it with parallel programming. For, converting the available algorithms into parallel, the subject expert is required to verify that the newly developed algorithm is working fine in the parallel programming. The researchers of the particular domain have to develop parallel algorithm for parallel programming, otherwise the output of the parallel algorithm may be change, or the output of the parallel program may not be correct. The parallel programming can be performed with parallel compiler. The parallel compiler can compile the program efficiently, and execute it in a parallel mode.

# Chapter 4

# Implementation and Testing

While Serial SOM training is relatively simple to implement, the Update neuron weights for each input pixel, it makes it necessary that all nodes(threads) communicate with each other at every iteration when needing the BMU for the current input pixel. for this reason Serial SOM training is slow in term of performance ,To speed up the SOM training, Kohonen proposed a batch version of the SOM called the Batch SOM , Moreover , the Batch SOM training is well-suited for parallelization. Batches of input data can be processed in parallel and neuron vectors can be updated at the end of iteration . This facts allows to significantly optimize network performance when processing high resolution images.
In this chapter we present the main steps of the implementation , The testing and results ,we will provide some interesting comparisons and we will discuss the available hardware and software platforms, which used for parallel processing.

## 4.1 Software Configuration

The Serial SOM application is written in C++ and does not use any platform specific operations. and the parallel application work on C++ OpenMP API. In this thesis, Ubuntu 17.10 Linux OS was used both for development and testing. For this reason the application works only in Linux. I decided to use this OS, because it is open, stable, secure, and free, it can be setup and run without a desktop manager, It saves resources and allows for independent testing with as little irrelevant processes as possible. It is also relatively popular among the scientific community.

### 4.1.1 Openmp

OpenMP is a parallel programming model targeted at homogeneous shared memory systems. It specifies a series of compiler directives, library functions and environment variables which can be used to express parallelism in Fortran and C/C++ programs. OpenMP handles parallelism by dispatching the execution of parallel code sections on multiple parallel threads,usually one thread per processor core. It is based on a fork-join model of computation. The sequential code sections are executed on a master thread. Upon entering a parallel section the OpenMP runtime forks execution on worker threads. When leaving a parallel section a synchronization barrier ensures all threads are done and execution continues on the master thread.
Parallelism is explicit, meaning the developer inserts compiler directives and library calls directly to the application code to control the parallelization. An OpenMP compiler interprets the OpenMP directives in the application code and applies the necessary transformations and runtime calls. The omp parallel directive defines a parallel region that encompasses the next code section. The number of threads can be specified by appending the $num_threads(n)$ option to the directive. Data Parallelism is specified by attaching a omp for directive to for loops inside a parallel region. The compiler extracts the loop body into a microtask and replaces it with a call to said microtask. Microtasks will be scheduled on all worker threads, which will execute a subset of the loop iterations, called chunks. The number of iterations in a chunk will depend on the particular scheduling chosen, or can be forced by the user. The schedule is determined by appending the schedule(type [, chunk size]) option to the directive, where type is the scheduling type and chunk size is an optional integer value. The five OpenMP loop schedules are:

- **Static** Allocates chunks statically to each worker thread in a cyclic way. The default chunk size is the number of loop iterations divided by the number of threads in the parallel region rounded up to the nearest integer.
- **Dynamic** Allocates chunks dynamically to each worker thread as they become available. The runtime needs some form of synchronization or atomic operations to determine the next chunk to process. The actual implementation is platform dependent. The default chunk size is one.
- **Guided** The guided schedule behaves like a static schedule where the chunk size is variable and depends on the amount of remaining iterations to process. It strikes a compromise by starting with larger chunks that show less scheduling overheads, and gradually decreasing the chunk size to improve load balance.
- **Auto** The compiler or runtime is free to determine the best schedule to use static, dynamic or guided.
- **Runtime** The schedule is determined according to the value of the OMP_SCHEDULE environment variable at runtime.

**Data Access** By default, OpenMP considers that variables declared outside the scope of a parallel region are shared among all threads. Variables declared in the parallel region areprivate to the thread. Access modifiers can be appended to a parallel region definition to specify if particular variables are shared or private, and to define if their values should be updated upon entering or leaving a parallel region. OpenMP also defines reduction operations that can be applied on private thread variables upon termination of a parallel construct.

```c
#include <stdio.h>
  int main (void) {
  int a, i;

  //Declare a parallel region
  #pragma omp parallel shared(a) private(i)
  {
    //Initialize accumulator variable 'a' on the master thread only
    #pragma omp master
    a = 0;

    //Synchronizes all threads
    #pragma omp barrier

    //Declare a data parallel for loop
    #pragma omp for schedule(static, 4) num_threads(8) reduction(+:a)
    for (i = 0; i < 256; i++) {
      a += i;
    }

    //Any single thread prints the result (implicit barrier)
    #pragma omp single
    printf("Sum is %d\n", a);
  }
}
```

Figure 4.1: example of an openmp application from implements a parallel sum.

Synchronization. The main OpenMP synchronization directives are :

- **omp barrier** Threads wait until other threads reach that particular point in the program before continuing.
- **omp master** The code section to which it is applied is executed only by the master thread.
- **omp single** The code section to which it is applied is executed by the a single thread, the first to reach it.
- **omp critical** Any code sections to which it is applied are protected and only one thread can execute any of the critical sections at a time. A name can be given to the critical section, in which case only one thread can execute any of the homonym critical sections. Parallel execution of critical sections with different names is allowed.
- **omp atomic** An update to a variable is guaranteed to be executed atomically.

## 4.2 Data format

The implemented feed network uses unsupervised learning. Therefore in the training phase ,the dataset it collection of Pixls, These are read from BITMAP file .
A 24-bit BMP file, then, is essentially just a sequence of bits, (almost) every 24 of which happen to represent some pixels color. But a BMP file also contains some "metadata," information like an images height and width. That metadata is stored at the beginning of the file in the form of two data structures generally referred to as headers , The first of these headers, called BITMAPFILEHEADER is 14 bytes long, The first 2 bytes of the BMP file format are the character "B" then the character "M" in ASCII encoding. A typical application reads this block first to ensure that the file is actually a BMP file .
and that it is not damaged. The second of these headers, called BITMAPINFOHEADER, is 40 bytes long. Immediately following these headers is the actual bitmap: an array of bytes, triples of which represent a pixels color.

| offset | type | name | |
|---|---|---|---|
| 0 | WORD | bfType | BITMAPFILEHEADER |
| 2 | DWORD | bfSize | |
| 6 | WORD | bfReserved1 | |
| 8 | WORD | bfReserved2 | |
| 10 | DWORD | bfOffBits | |
| 14 | DWORD | biSize | BITMAPINFOHEADER |
| 18 | LONG | biWidth | |
| 22 | LONG | biHeight | |
| 26 | WORD | biPlanes | |
| 28 | WORD | biBitCount | |
| 30 | DWORD | biCompression | |
| 34 | DWORD | biSizeImage | |
| 38 | LONG | biXPelsPerMeter | |
| 42 | LONG | biYPelsPerMeter | |
| 46 | DWORD | biClrUsed | |
| 50 | DWORD | biClrImportant | |
| 54 | BYTE | rgbtBlue | RGBTRIPLE |
| 55 | BYTE | rgbtGreen | |
| 56 | BYTE | rgbtRed | |
| 57 | BYTE | rgbtBlue | RGBTRIPLE |
| 58 | BYTE | rgbtGreen | |
| 59 | BYTE | rgbtRed | |
| ... | | | |
| 243 | BYTE | rgbtBlue | RGBTRIPLE |
| 244 | BYTE | rgbtGreen | |
| 245 | BYTE | rgbtRed | |

Figure 4.2: the structure of the BMP Image on disk according to Microsoft.

## 4.3 SOM Training

**Algorithm 1:** Serial SOM RGB color space

**Data:** A bitmap Image.
**Result:** A segmentation of the input image.
**begin**
    Initialize the map with random values
    **for** *each epoch (from t to T)* **do**
        **for** *each image pixel* $\varphi_p[R_p, G_p, B_p]$ **do**
            Compute for each neuron the Euclidean distance $d_j$ with 3.1
            Find the index $j'$ such that $d_{j'}$ is a minimum with 3.2.
            **for** *each neuron* **do**
                Update the weight vectors with equation 3.3
        reduce the learning rate using $\alpha(t)$
        reduce the radii that define the topological neighborhoods $r(t)$ with 3.6
        Compute $h_{ck}(t)$ with 3.5
// *Classification of pixels*
**for** *Each image pixel* $\varphi_p[R_p, G_p, B_p]$ **do**
    Compute for each neuron the Euclidean distance $d_j$ with 3.1
    Find the index $j'$ such that $d_{j'}$ is a minimum with 3.2.
    $\varphi_p \longleftarrow W_{j'}$

**Algorithm 2:** Parallel Batch SOM

```
Data: A bitmap Image.
Result: A segmentation of the input image.
begin
    Initialize the map with random values
    Partitions set of input vectors(images pixels)
    Send for each task subset of input data
    for neurons in the grid do
        GlobalNumerator ⟵ 0
        GlobalDenominator ⟵ 0
    for epoch = 1 to N_epochs do
        for neurons in the grid do
            LocalNumerator ⟵ 0
            LocalDenominator ⟵ 0
        Coordinator calculates new σ
        for each image pixel φ_p[R_p, G_p, B_p] do
            for neurons in the grid do
                Find the neuron with minimum distance from the
                 instance φ_p.the winner
            for neurons in the grid do
                Accumulate local numerator and denominator in eq.3.7
        Critical section begin
            Add local numerator and denominator to
             GlobalNumerator and GlobalDenominator
            End Critical section
        Barrier (tasks wait until all other tasks reach that particular
         point)
        Single (The code section to which it is applied is executed by
         the a single task) begin
            for neurons in the grid do
                Update the weight vectors
            for neurons in the grid do
                GlobalNumerator ⟵ 0
                GlobalDenominator ⟵ 0
            End single
        Barrier (tasks wait until all other tasks reach that particular
         point)
// Classification of pixels
for Each image pixel φ_p[R_p, G_p, B_p] do
    Compute for each neuron the Euclidean distance d_j with 3.1
    Find the index j' such that d_j' is a minimum with 3.2.
    φ_p ⟵ W_j'
```

## 4.4 Testing

In this section .we will test all the implementation models of the SOM neural network , and we will describe the hardware and software we used for testing then we will summarize the results of network implementation.

### 4.4.1 Testing environment

While working on the implementation ,I was using one machine for development and test

**Hardware specification**

**CPU**

- Intel Core i3-4005U @ 1.7 GHz
- 2 Core 4 thread
- 64bit instruction set
- L1 Data Cache Size 2 x 32 Kbytes
- L1 Instructions Cache Size 2 x 32 KBytes
- L2 Unified Cache Size 2 x 256 KBytes
- L3 Unified Cache Size 3072 Kbytes
- Technology 22nm

**RAM**

- 4GB DDR3 800MHZ SODIMM by Hynix/Hyundai
- Single channel
- actual frequency 798Mhz

### 4.4.2 Testing methodology

I will test the implementation of parallel SOM ,the aim of testing is to evaluate quality of clustering and to assess the performance of the implementation.

**Experiment to Evaluate quality of clustering** Goal here is evaluate quality of clustering, The tests are performed using images of the Berkeley Segmentation Database (BSD), which is becoming the benchmark to test algorithms related to color image segmentation . The BSD contains 300 color images and the size of the images is 481 X 321 or 321 X 481. we selected 16 images of the BSD to perform the experiments . our implementation of parallel SOM include two model one with RGB as color space and other with HSV color space.

**Batch SOM with RGB as color space** this implementation deal with the problematique of data dimensionality reduction(reduce the number of color without match losing Quality of the image ) ability to overcome some critical problems such as image compression.

**The training process** The ability to learn is the key concept of neural networks. The aim of the process is to find the optimal parameters of the network for, solving the given task. Before the training process starts, network parameters need to be initialized. Initial values of the weight are chosen randomly, Learning is then carried out on the training set by feeding the training data through the network. It is an iterative process, where the outputs produced on each input from the training set are analyzed and the network is repeatedly being adjusted to produce better results. The network is considered to be trained after reaching the number of iteration.

Batch SOM configuration with RGB as color space (in batch SOM there is no learning rate :

- MAPSIZE 7 X 7 (49 neurons)
- 50 Iteration

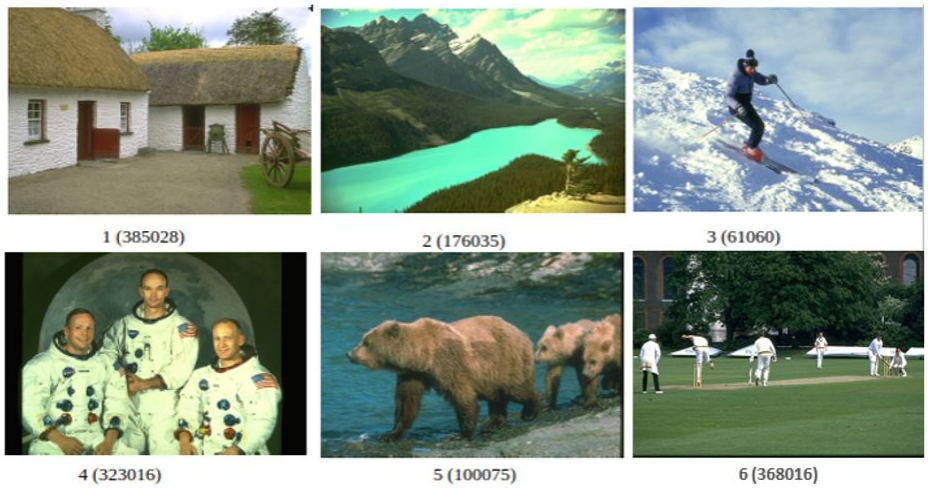



Figure 4.3: Images of the BSD employed for experiments.

**Performance** To evaluate the performance we need to compare the parallel batch SOM with sequential SOM ,we train the different network with the same image (1920 X 1280) and the number of epoch was set to 50 ,the learning rate for sequential SOM is 0.9 The neuron for the SOM were arranged in a rectangular grid pattern.

### 4.4.3 Results

**evaluate the quality of segmentation** In order to evaluate the quality of segmentation we looking the histogram of the original image and segmented image for all test images, It is easy to appreciate the reducing of the color of the results images , and the obtained images still preserved the finer details of the input image. see input images 4.3 ,output images 4.4 fig(4.7 ,4.8 ,4.9 ,4.10 ,4.11 ,4.12)

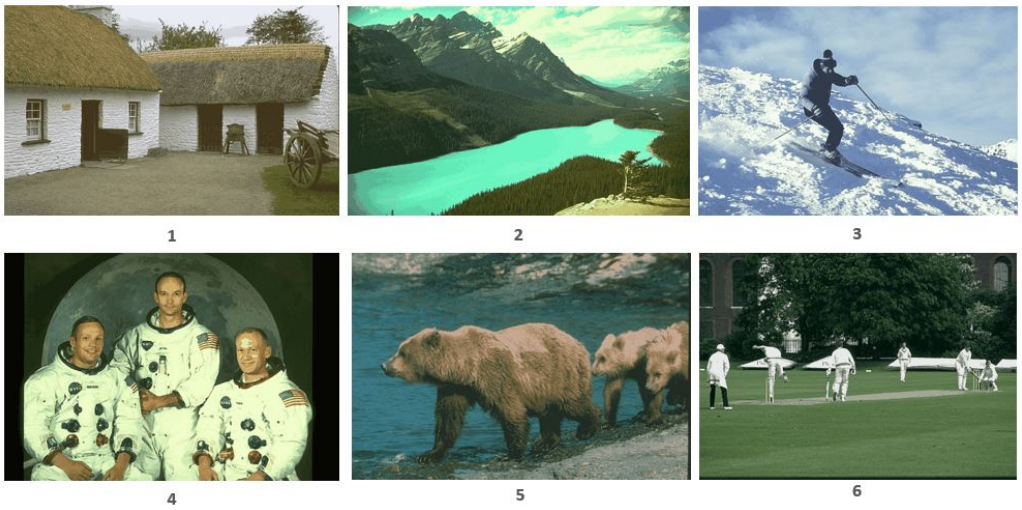



Figure 4.4: Images obtained by using the batch SOM with 7 X 7 neurons RGB as a color space

### Performance Analysis

figure 4.6 summarize all performance results , it is possible to see that the maximum computational performance done by 2 processing elements compared with resource allocated (2 virtual processer ),in 3 and 4 processing elements there is slow improvement of the performance because of limited bandwidth of the global memory with just a single channel , as well as the time spent waiting for critical section and on synchronization barriers.

Table 4.1: execution time of varying batch SOM thread and traditional sequential SOM

| | 5X5 | 6X6 | 7X7 | 8X8 | 9X9 | 10X10 | 11X11 | 12X12 | 13X13 |
|---|---|---|---|---|---|---|---|---|---|
| Sequential SOM | 239 | 344 | 461 | 605 | 752 | 992 | 1111 | 1341 | 1533 |
| batch SOM with 1 thread | 236 | 336 | 458 | 593 | 747 | 923 | 1117 | 1326 | 1561 |
| batch SOM with 2 thread | 121 | 173 | 240 | 304 | 386 | 473 | 572 | 684 | 800 |
| batch SOM with 3 thread | 114 | 161 | 219 | 285 | 356 | 439 | 530 | 632 | 739 |
| batch SOM with 4 thread | 101 | 143 | 191 | 251 | 312 | 384 | 466 | 553 | 649 |

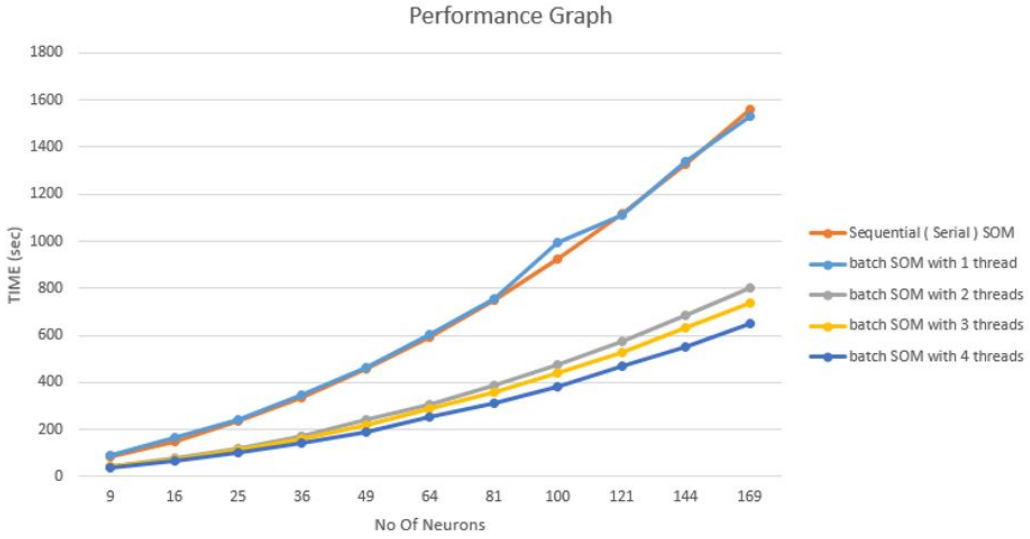



Figure 4.5: Visualization of table 4.1

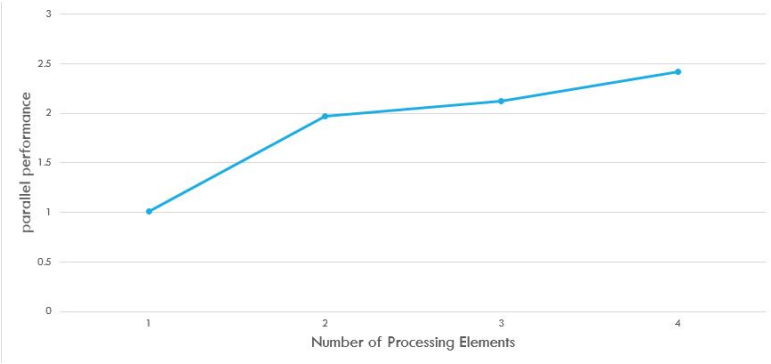



Figure 4.6: parallel performance compared with traditional sequential SOM

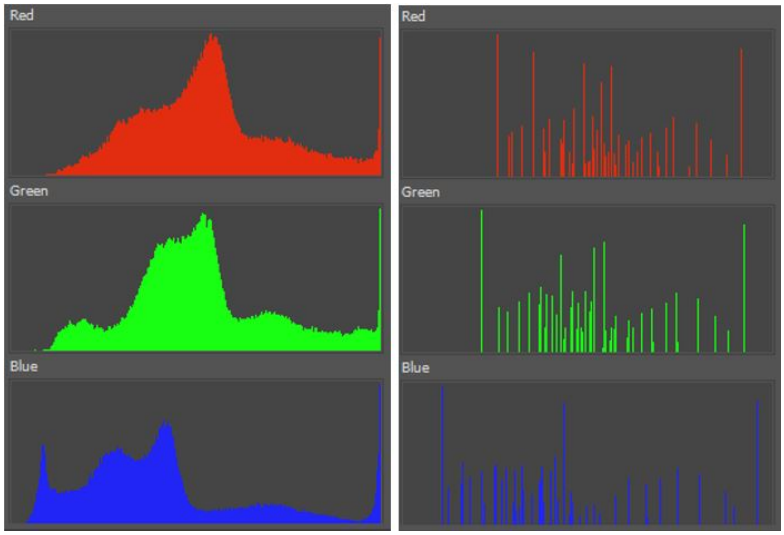



Figure 4.7: Img $N°1$ left:Histogram of original image — right: Histogram of segmented image

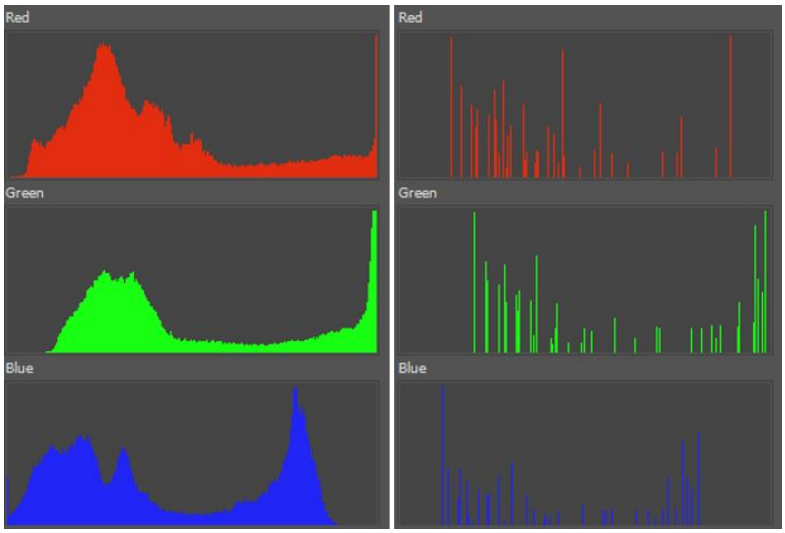



Figure 4.8: Img $N°2$ left:Histogram of original image — right: Histogram of segmented image

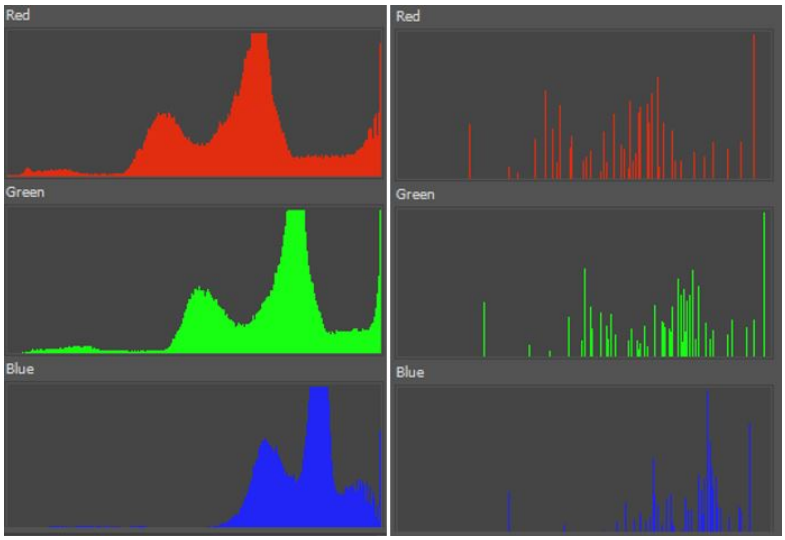



Figure 4.9: Img $N°3$ left:Histogram of original image — right: Histogram of segmented image

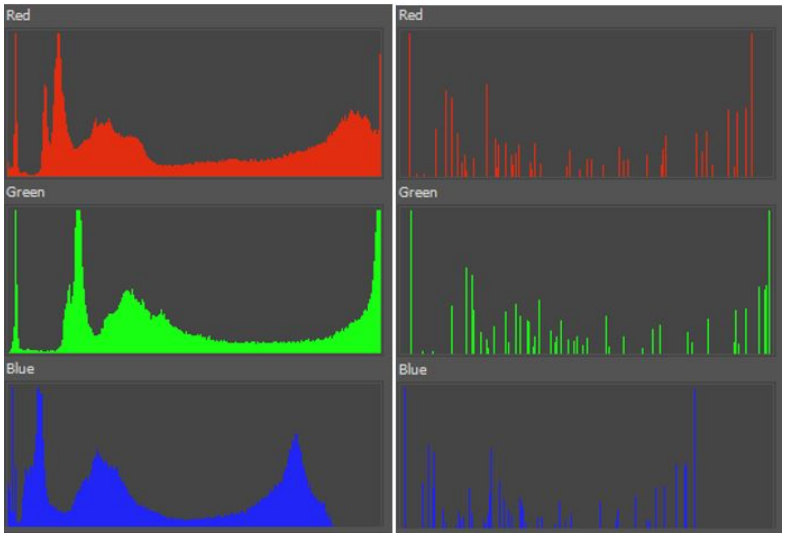



Figure 4.10: Img $N°4$ left:Histogram of original image — right: Histogram of segmented image

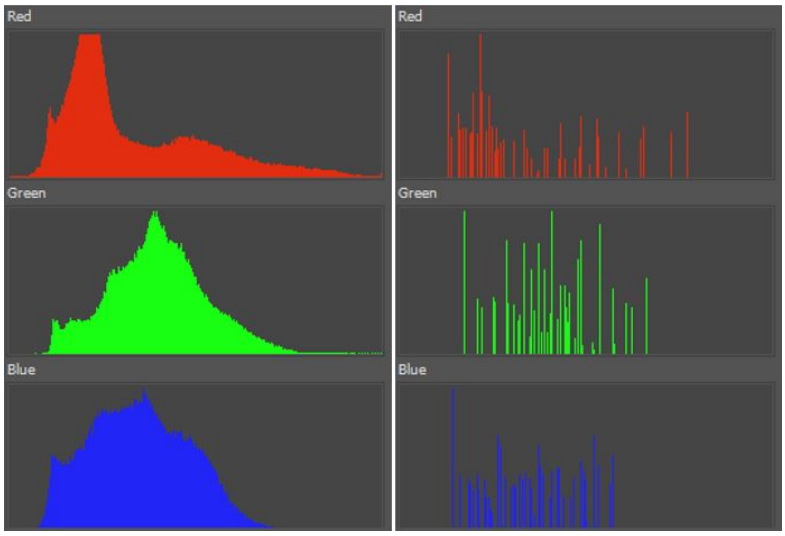



Figure 4.11: Img $N°5$ left:Histogram of original image — right: Histogram of segmented image

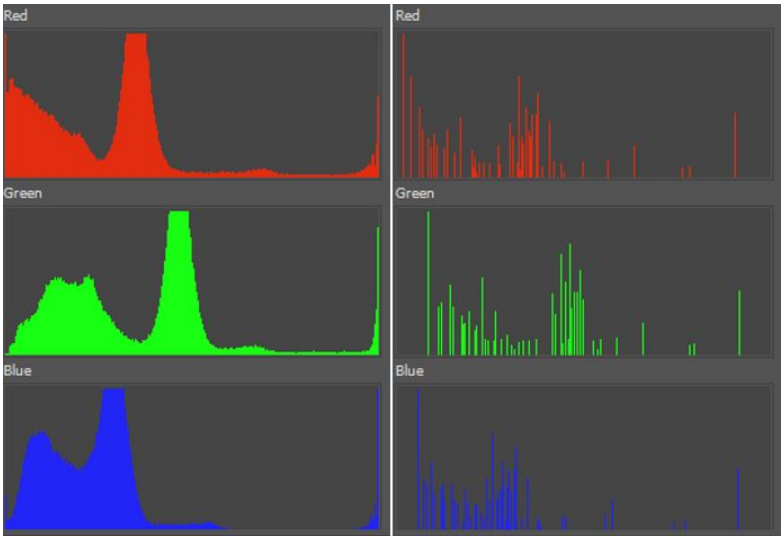



Figure 4.12: Img $N°6$ left:Histogram of original image — right: Histogram of segmented image

# Chapter 5

# Conclusion

Like other neural network algorithm the Kohonen SOM is popular and effective machine learning technique , great progress has been made parallelizing the expensive training phase of the SOM .this thesis attempts to .provide parallel implementation of Kohonen SOM neural network for segmentation of color image. This was achieved using c++/openmp .
One goal of the project was to evaluate how well parallel SOM to preserve the finer details of the image The concrete results of this project demonstrate that the implementation satisfies this goal.
The other goal was to evaluate the performance in order to reduce processing times ,in this case the implementation is analyzed for a varying number of thread , the results are encouraging .

# Bibliography

[1] J. Arfan, I. Muhammad, H. Ayyaz and M. Anwar, "Wavelet-Based Color Image Segmentation using Self-Organizing Map," in International Conference on Computer Engineering and Applications, 2011.

[2]CS231n Convolutional Neural Networks for Visual Recognition, stanford, [Online]. Available: http://cs231n.github.io/neural-networks-1/intro. [Accessed 23 5 2018].

[3] MADELINESCHIAPPA, Man vs machine: comparing artificial and biological neural networks, Sophos News, 21 09 2017. [Online]. Available: https://news.sophos.com/en-us/2017/09/21/man-vs-machine-comparing-artificial-and-biological-neural-networks/. [Accessed 24 5 2018].

[4] C. ,. L. Fernando, EXPLORATORY GEOSPATIAL DATA, p. 158, 2005.

[5] S. ,. G. Navdeep, Overview and Applications of Artificial Neural Networks," xenonstack, 5 5 2017. [Online]. Available: https://www.xenonstack.com/blog/data-science/overview-of-artificial-neural-networks-and-its-applications. [Accessed 23 5 2018].

[6] D. Manasi, NATURAL SCENE SEGMENTATION BASED ON INFORMATION FUSION, p. 63, 2005.

[7] Moeslund, Image Acquisition, in introduction to video and image processing, springer, 2012.

[8] T. Seemann, Digital Image Processing, Monash University, 2002.

[9] F. Garca-Lamont, A. Cuevas and Y. Nio, Segmentation of color images by chromaticity, IngenIera eInvestIgacIn vol. 36(2) , p. 12, 2016.

[10] M. Dubravko, Brief Review of Self-Organizing Maps, p. 6, 2017.

[11] S. Bruno and Marques, "A Hybrid Parallel SOM Algorithm for Large," p. 11.

[12] M. ,. M. Ameya, "Extending the Growing Hierarchical Self Organizing Maps for a Large Mixed-Attribute Dataset Using Spark MapReduce," p. 99, 2015.

[13] wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Parallel_computing. [Accessed 25 5 2018].

[14] C. AMZA, REVIEW ON NEURAL NETWORK-BASED IMAGE, De Montfort University Mechanical and Manufacturing Engineering . 23.

[15] X. M. PUJOL, Image Segmentation Integrating Colour, Texture and Boundary Information.

[16] image segmenation, Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Image_segmentation. [Accessed 17 9 2018].

[17] M. Leung, "Kohonen Self-Organized Maps," mleung@poly.edu, 2007.