Major Project Report
On

FUZZY IMPULSE NOISE REMOVAL IN VIDEO SEQUENCES USING BACTERIAL FORAGING ALGORITHM

Submitted in partial fulfillment of the requirements

For the award of the degree of

Master of Technology
In
Information Systems

Submitted By:

TULIKA BANSAL

Roll No. 17/ISY/2K10

Under the Guidance of

Dr. O. P. Verma

(HOD, IT Department)

Department of Information Technology

DELHI TECHNOLOGICAL UNIVERSITY

Bawana Road, Delhi-110042

2010-2012

CERTIFICATE

This is to certify that the work contained in the thesis titled "***Fuzzy Impulse Noise Removal In Video Sequences Using Bacterial Foraging Algorithm***" is an original piece of work which has been carried out by **Tulika bansal (**17/ISY/2K10**)** under my supervision. This work has not been submitted elsewhere for a degree.

Dr. O.P.Verma

HOD (IT Dept.)

Department of Information Technology

Delhi Technological University, Delhi

June 2012

ABSTRACT

This thesis presents a new optimized fuzzy filter for denoising of color video sequences corrupted with random impulse noise. Optimization is done using Bacteria Foraging Algorithm (BFA). RGB color model is used to represent the frames of the color video. Two successive steps of filtering are used to remove the noise from the video. The first step detects the noisy pixels along with the amount of noise present and are then corrected using the median of the noise-free pixels. The output of the first step acts as an input to the second filtering step and further refines the result to give the final output. The filter presented in this paper is a 3D spatio-temporal filter that considers spatial, temporal as well as the color information. A pixel in one color component is compared with its neighboring pixels within the same frame, with the corresponding pixels in neighboring frames and also with the pixels in other two color components. Six fuzzy membership functions are defined which are then applied to a fuzzy rule. The fuzzy rule decides whether the given pixel is noisy or noisefree. Mean Square Error (MSE) is used as an objective function, which is optimized using the bacterial foraging algorithm to learn the parameters of membership functions. Finally, the correction term for each color component of each frame is calculated which is added to the original noisy frame component to remove the noise. Only noisy pixels are corrected while the noisefree pixels are left untouched. The experimental result on several color video sequences proves that the impulse noise is efficiently removed by the proposed fuzzy filter.

Acknowledgements

I would like to express my sincere gratitude to my advisor, Dr. O.P.Verma for giving me an opportunity to work with him and for his advice, encouragement and constant support. I wish to thank him for extending me the greatest freedom in deciding the direction and scope of my research. It has been both a privilege as well as a rewarding experience working with him.

I would also like to thank all my friends and colleagues here at Delhi Technological University for their valuable comments and suggestions, many of which have been vital to this work. Furthermore, I want to thank the faculty of Department of Information Technology and the staff for providing me the means to complete my degree.

I would like to thank my parents for their love, sacrifice, understanding, encouragement and support. I am forever indebted to them for all that they have given me. Finally, special thanks to my brother for his love, affection and help.

TULIKA BANSAL

Roll No. 17/ISY/2K10

M.Tech (Information Systems)

Department of Information Technology

Delhi Technological University

Bawana Road, Delhi-110042

CONTENTS

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

1.1 Noise

The signal is the planned, ordered information and important part of the data that we want to measure. Noise on the other hand is the unwanted, unexpected and unstructured information. Noise is random information and unpredictable variations in the measured signal. The term noise usually means an undesirable random disturbance conflicting with the desired signal.

Image noise is a random, unwanted variation in brightness or color information in an image. It is undesirable and extraneous information that results in wrong pixels values. Image noise can originate in film grain, or in electronic noise in the input device (scanner or digital camera) sensor and circuitry, or in the unavoidable shot noise of an ideal photon detector. Incorrect image acquisition and transmission are the major two factors responsible for introduction of noise in images.

Although there are many types of image noises [1-2], Gaussian noise and salt-and-pepper noise are the major two. The current work focuses on removal of salt-and-pepper noise from videos using fuzzy filter. Fuzzy filters provide promising result in image-processing tasks that cope with some drawbacks of classical filters. Fuzzy filter is capable of dealing with vague and uncertain information. Sometimes, it is required to recover a heavily noise corrupted image where a lot of uncertainties are present and in this case fuzzy set theory is very useful. Each pixel in the image is represented by a membership function and different types of fuzzy rules that considers the neighborhood information or other information to eliminate filter removes the noise with blurry edges but fuzzy filters perform both the edge preservation and smoothing [3].

Salt and pepper noise is a form of noise typically seen on images. It represents itself as randomly occurring white and black pixels. An image containing salt-and-pepper noise will have dark pixels in bright regions and bright pixels in dark regions. This type of noise can be caused by analog-to-digital converter errors, bit errors in transmission, etc. For images corrupted by salt-and-pepper noise, the noisy pixels can take only the maximum and the minimum values in the dynamic range. An effective noise reduction method for this type of noise involves the usage of a median filter or a contra harmonic mean filter. Salt and pepper

noise creeps into images in situations where quick transients, such as faulty switching, take place.

For removing noise from images, a number of fuzzy filters have been used in the past [4]. These include FIRE filters [5-6], hybrid filters [7-10], fuzzy vector directional filters [11], fuzzy median filters [12], and fuzzy stack filters [13] to name a few.

The mean filter or the average filter helps in smoothing operations. It suppresses the noise that is smaller in size or any other small fluctuations in the image. Smoothing or averaging operation blurs the image and does not preserve the edges. These are not used in removing noise spikes. Adaptive weighted mean filter is similar to mean filter where the gray level is replaced by a weighted average of the gray values. Weights are calculated from the gray-level difference. If this difference exceeds a certain threshold, then the pixel is a noise pixel. Adaptive Weiner filter replaces the center value of the pixel by sum of the local mean value and a fraction of contrast, where this fraction depends on the local estimation of the variance [14]. Median filter is found to be the most effective non linear filter for removing impulse noise but for higher noise level, median filters lead to loss of details and lead to blurring [15]. Although multistate median filter [16], and adaptive median filters [17] provided solution to the drawback of median filters but these filters were also not able to provide satisfactory results at higher noise level.

1.2 Prior Work in Video

In this work we focus on removal of impulse noise from the video frames. However a number of techniques have been developed in past to remove noise from image sequences, most of them are either unable to provide desirable results or provide results only with low noise presence.

In this section we present an overview of the existing approaches for image sequence denoising. A comparison between different methods is given in terms of their framework, denoising performance and complexity.

With the advance in technology, capturing of videos has become much easier in today's time but there are still a number of factors due to which impulse noise gets introduced into the

frames of the videos. Unprofessional acquisition and imperfect transmission are the major two factors which lead to noise introduction in to the video frames. Presence of noise in image sequences does not only hampers its visual quality but also effects the performance of a number of processing tasks to be performed on video such as coding, analysis or interpretation [18]. Therefore, filtering algorithms are necessary to remove the noise from the frames as well as to improve the performance of these processing tasks.

Spatial filters take into account only the current frame and remove noise from it independent of the neighboring frames. The static filters are obtained by extending the 2D image filter support to the 3D time-space support. We call them static because they do not take into account the dynamic character of image sequences. A drawback of this approach is that the strong temporal correlation in image sequences is not used. In addition, temporally inconsistent results may be obtained, causing annoying artifacts. An advantage is that additional motion blurring is avoided. Although purely spatial methods for image sequence filtering will be often used in hardware implementations, they are hardly reported in scientific literature. The most classical example is the arithmetic mean. Mean filter averages all the pixels in the neighborhood of the current pixel and thus results in blurring of the edges as well as blurring of the moving regions. Furthermore, it removes many details. Lee [19] statistical correction performs a more conservative estimate when a large variance of the mean is observed. As a consequence, the edges and boundaries between moving regions are kept noisy. The neighborhood (or sigma) filters [20, 21] avoid the blurring effect of the mean filter. Neighborhood filters are based on the assumption that pixels belonging to a same region have a similar grey level value. Therefore, one should restrict the average to pixels with a small grey level difference with the one in restoration. The neighborhood filter is a better denoising tool than Lee's correction. It maintains sharp boundaries, since it averages pixels belonging to the same region as the reference pixel. Unfortunately, this method fails when the standard deviation of the noise exceeds the contrast of edges. The neighborhood filter does not blur the image as it averages only pixels with a similar grey level but many isolated noisy points are still visible. All of the three above mentioned classes of static filters are not adapted to the removal of local artifacts such as the "dirt and sparkle". The mean filter reduces these artifacts because it averages all the pixels inside the spatiotemporal support. The cost of this reduction is the blurring of boundaries and details. Since the mean performed at local artifacts has a large variance, the Lee statistical correction

performs a more conservative estimate and keeps the original values. Thus, it maintains dirt and sparkle. Since the neighborhood filters average pixels with a similar grey level value, they also keep these artifacts. The median filter [22] chooses the median value, that is, the value which has exactly the same number of grey level values above and below in a fixed neighborhood. This filter is optimal for the removal of impulse noise on images. This algorithm is able to reduce the additive noise and to remove as well dirt and sparkle. The median filter preserves the main boundaries, but it tends to remove the details and to blur the boundaries of moving regions. The median filter preserves the edges and at the same time reduces the noise and the local artifacts. However, the result is blurred because of the poor adaptation to movement. In all these spatial filters, the temporal correlation between the neighboring frames is ignored and hence is not a good approach for removing noise from a video. Even the most advanced spatial denoisers, such as Wiener [23] and wavelet filtering [24], cannot deliver good video denoising results. Therefore pure spatial denoisers methods are not appropriate for video. Temporal filters woks only on the temporal correlation and neglect the spatial correlation. Spatiotemporal filters are those filters which take into account both the spatial as well as temporal correlation and remove noise from a video by considering current frame as well as previous and next frame. A survey of several temporal and spatio-temporal filtering schemes to reduce the noise in image sequences can be found in [18]. Filtering schemes are applied on the video depending on the static and dynamic nature of image sequences. Denoising techniques need to take care of the difference between the noisy pixels and motional pixels while removing noise from a video. Averaging temporal pixels to smooth the impulse noise works fine in case of non moving objects and non moving scenes but produces blurring effects and ghosting artifacts in presence of moving objects and moving scenes. Adaptive algorithms and the motion compensation are the two techniques that take care of the dynamic nature of video sequences and are used to avoid this blurring effect. Adaptive algorithms implicitly take into account the motion assumption in the design of the method. Motion compensation algorithms compute motion estimation as a preprocessing step. A brief description of these techniques is explained in [25]. The adaptive filters [26-28] are designed specifically to deal with image sequences and take into account the possibility of a motion. The aim of these filters is to avoid the blurring effect where motion occurs. Adaptive filters take into account the dynamic character of image sequences but do not compute explicitly the optical flow. Adaptive filters may also be preferred when speed is

required or the sequence is highly corrupted. The motion compensated filters [29] estimate explicitly the motion of the sequence by a motion estimation algorithm. The motion compensation filters have a high computational cost and the accuracy of motion estimation algorithms decreases with noise. However, when dealing with moderated noise, motion compensation filter are usually preferred. In fact, many papers dealing with adaptive filters conclude that motion compensation is required to overcome the temporal artifacts. The availability of accurate motion estimation is crucial for the performance of the motion compensation filters.

The first filters for video denoising were single resolution filters. These were often some well-known 2D filters extended to a spatio-temporal neighborhood. Some examples are the 3D KNN-filter [30-32] and the 3D threshold averaging filter [20, 32], which try to preserve the details by averaging only over the k nearest neighbors (KNN) and the neighbors lying within a certain distance from the given pixel value respectively. More recent extensions of these filters, that are made more adaptive to a local spatio-temporal neighborhood are e.g. the motion and detail adaptive KNN-filter [33] and the multiple lass averaging filter [34, 35]. Another well-known single resolution method is the 3D rational filter [36], where the filtered output for a pixel is determined as a rational function of the grey values in a spatio-temporal neighborhood. Other recent single resolution filters can e.g. be found in [37, 38]. Both filters take into account pixels from neighboring frames in the averaging, which not necessarily are the pixels at the same spatial position, but the estimated corresponding objet pixels which possibly have been displaced due to motion between frames.

1.3 Motivation and Proposed Work

Most of the filters developed for video are 2-D spatial filters that filter each of the frames of the video successively. They consider only one frame at a time and remove the noise. However by considering only one frame at a time they neglect the temporal relation between frames of the video. This thesis presents a spatio- temporal filter that removes the noise from the video using fuzzy logic. The filter calculates the degree with which the pixel is noisy and removes the noise from it accordingly. Hence, only the noisy pixels are corrected and uncorrupted pixels are left

untouched. Since the filter removes noise from the color video, only spatial and temporal information are not enough, color information in other bands is required as well. Hence, the noise is corrected by considering the current frame, the neighboring frames and the values from the other color components. The six membership functions namely; Big, Dissimilar, Terminal, High, Variant and Temporal are defined. Big, Dissimilar and Terminal membership function gives spatial information about a given pixel. High, Variant and Temporal membership function give temporal information about the pixel. A Fuzzy rule is then applied on each pixel to check whether it is noisy or noise-free. If the pixel is found to be noisy, it is replaced by the median value of the noise-free pixels in their neighborhood otherwise it is not changed. The unique characteristic feature of this filter to exploit the spatial correlation and temporal correlation along with color components in a 2 step process makes it work even with high impulse noise presence. Since only noisy pixels are changed and noise free pixels are left untouched, blurring is minimized. The parameters involved are optimized using the Bacteria Foraging Algorithm to provide with best results. We show the video denoising performance in terms of peak signal-to-noise ratio (PSNR) and the experimental results on several videos proved the efficacy of this fuzzy filter as compared to other well known video filters.

1.4 Thesis Outline

Chapter 2 gives a short overview of the basic concepts in fuzzy set theory and fuzzy sets in image processing. Chapter 3 then presents the definition of the impulse noise, which is considered in this paper, an algorithm to compute the median and the proposed spatio-temporal filtering steps to reduce the impulse noise. Parameter training using Bacteria Foraging and parameter initialization is described in chapter 4. Chapter 4 also presents the value of parameters obtained after the optimization and describes the effect of parameter variation. Results and discussion and comparison with other filters are presented in chapter 5. Finally, conclusions and future work are given in chapter 6.

CHAPTER 2

FUZZY LOGIC

2.1 Introduction to fuzzy set

The term fuzzy means vagueness. Fuzziness occurs when the boundary of a piece of information is not clear cut. The basic idea of the fuzzy set theory is that an element belongs to a fuzzy set with a certain degree of membership. Thus, a proposition is not either true or false, but may be partly true (or partly false) to any degree. Classical set theory or crisp theory allows the membership of elements in the set in binary terms – an element either belongs to or does not belong to the set. Classical set is a set with crisp boundary. They do not reflect the nature of human concepts and thoughts which tends to be abstract and imprecise. The flaw comes from the sharp transition between inclusion and exclusion in the set. Fuzzy set is a set without a crisp boundary. Transition from 'belong to a set' to 'not belong to a set' is gradual and smooth. Classical set has precise questions and precise answers whereas in fuzzy logic, the questions are not precise so, the answers cannot be given by classical set theory. The answers are also imprecise and can be given by fuzzy logic. Fuzzy sets have introduced by Lotfi A. Zadeh (1965) [39] as an extension to the classical notion of set.
The essential characteristics of fuzzy logic as founded by Zadeh Lotfi are as follows.

- In fuzzy logic, exact reasoning is viewed as a limiting case of approximate reasoning.
- In fuzzy logic everything is a matter of degree.
- Any logical system can be fuzzified.
- In fuzzy logic, knowledge is interpreted as a collection of elastic or, equivalently , fuzzy constraint on a collection of variables

Fuzzy logic uses the whole interval between 0 (false) and 1 (true) to describe human reasoning. In fuzzy sets, the smooth transition is characterized by membership functions that give fuzzy set flexibility. A fuzzy set is any set that allows its members to have different degree of membership, called membership function, in the interval [0, 1]. Membership function maps each element of a set to a membership grade between 0 and 1 to express the degree to which the element belongs to the set.

Fuzzy systems are very useful in two general contexts:

1. In situations involving highly complex systems whose behaviors are not well understood.
2. In situations where an approximate, but fast, solution is warranted.

2.2 Fuzzy sets in Image Processing

The concept of fuzzy logic has been extended in recent years to image processing by Hamid Tizhoosh and others at the Pattern Analysis and Machine Intelligence (PAMI) research group at the University of Waterloo (Waterloo, ON, Canada). There are various techniques for processing an image such as linear scaling, optical methods, digital processing and fuzzy techniques. Processing techniques such as linear scaling, optical methods have not been able to handle the disturbances occurring in processing an image. Fuzzy techniques used in processing an image have evolved as the most efficient solution for this problem. These techniques with fuzzy sets give much-improved image compared to the others.

Fuzzy image processing is the collection of different fuzzy approaches to image processing. These approaches understand, represent and process the images, their segments and features as fuzzy sets. The representation and processing depend on the selected fuzzy technique and on the problem to be solved.

Fuzzy set theory and fuzzy logic provide powerful tools to represent and process human knowledge in form of fuzzy if-then rules. Image processing poses many problems owing to imprecise and uncertain data, activities and results. However, randomness is not only the cause of this uncertainty, inherent ambiguity and vagueness of image data also contributes to this uncertainty. Beside randomness, which can be managed by probability theory three other kinds of imperfection in image processing are grayness ambiguity, geometrical fuzziness, and vague knowledge of image features.

Three main stages necessary to perform image processing using fuzzy logic are:

1. Image Fuzzification
2. Modification of membership values
3. Image Defuzzification

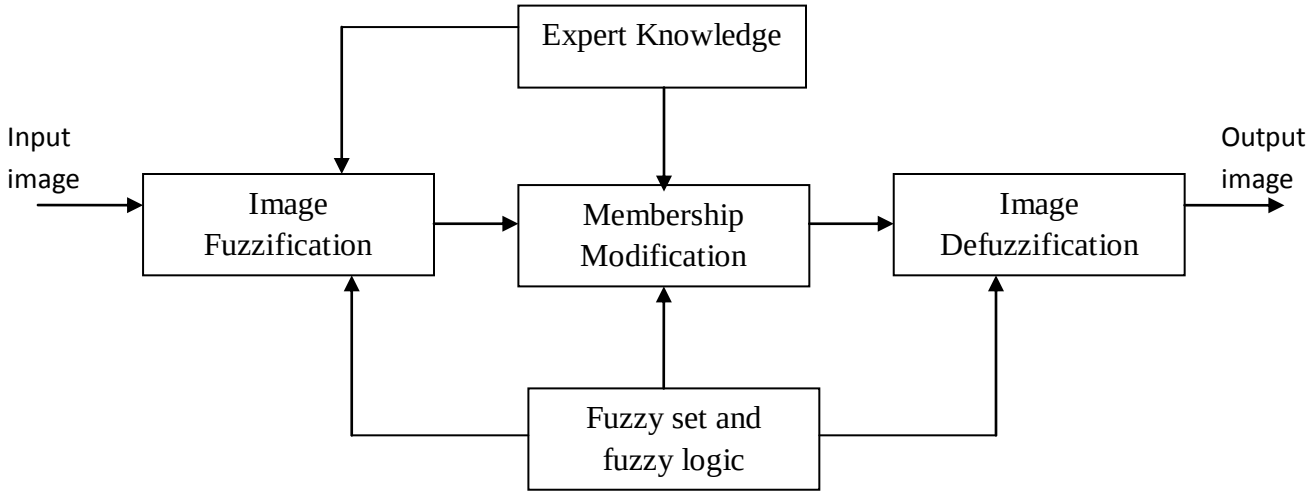


Figure 2.1 Fuzzy Image Processing [40]

Image Fuzzification is used to modify the membership values of a specific data set or image. After the image data are transformed from gray-level plane to the membership plane using fuzzification, appropriate fuzzy techniques modify the membership values. This can be a fuzzy clustering, a fuzzy rule-based approach, or a fuzzy integration approach. Decoding of the results, called defuzzification, then results in an output image. The main power of fuzzy image processing is in the modification of the fuzzy membership values.

2.3 Fuzzy Inference System (FIS)

Fuzzy inference system (FIS) is a method, based on the fuzzy theory, which maps the input values to the output values. The mapping mechanism is based on Membership Functions, Logical Operations and If-Then Rules. A Fuzzy Inferencing System (FIS) is one that uses fuzzy sets to make decisions or draw conclusions. Fuzzy inference systems have been successfully applied in fields such as automatic control, data classification, decision analysis, expert systems, and computer vision. Because of its multidisciplinary nature, fuzzy inference systems are associated with a number of names, such as fuzzy-rule-based systems, fuzzy expert

systems, fuzzy modeling, fuzzy associative memory, and fuzzy logic controllers, and simply (and ambiguously) fuzzy systems. The typical scheme consists of the following steps:

1. Fuzzificaton
2. Applying rules
3. Logic sum and defuzzyfication

The two most commonly used fuzzy inference systems are Mamdani-type and Sugeno-type. Mamdani-type was proposed in 1975 by Ebrahim Mamdani [41] as an attempt to control a steam engine and boiler combination by synthesizing a set of linguistic control rules obtained from experienced human operators. Mamdani‘s effort was based on Lotfi Sade‘s 1973 paper on fuzzy algorithms or complex systems and decision processes [42]. The most fundamental difference between Mamdani-type FIS and Sugeno-type FIS is in the way the crisp output is generated from the fuzzy inputs. While Mamdani-type FIS uses the technique of defuzzification of a fuzzy output, Sugeno-type FIS uses weighted average to compute the crisp output. But Sugeno has better processing time since the weighted average replace the time consuming defuzzification process. Mamdani-type inference expects the output membership functions to be fuzzy sets while Sugeno-type systems can be used to model any inference system in which the output membership functions are either linear or constant.

Mamdani method is widely accepted for capturing expert knowledge. It allows us to describe the expertise in more intuitive, more human-like manner. However, the expressive power and interpretability of Mamdani output is lost in the Sugeno - FIS since the consequents of the rules are not fuzzy. Mamdani-type FIS is not computationally efficient whereas Sugeno method is computationally efficient and works well with optimization and adaptive techniques, which makes it very attractive in control problems, particularly for dynamic non linear systems. Due to the interpretable and intuitive nature of the rule base, Mamdani-type FIS is widely used in particular for decision support application. Other differences are that Mamdani FIS has output membership functions whereas Sugeno FIS has no output membership functions. Mamdani FIS is less flexible in system design in comparison to Sugeno FIS as latter can be integrated with ANFIS tool to optimize the outputs [43].

CHAPTER 3

PROPOSED APPROACH

3.1 Impulse Noise

The proposed work is intended for the color video corrupted by random impulse noise. This implies that a pixel of a given color component is corrupted with noise by some probability p. The neighboring pixels of the given color component as well as pixels of the other color component can be noisy or noisefree independent of this pixel. Mathematically, noisy sequence f_n is obtained as follows:

$$f_n(x,y,z,t) = \begin{cases} f_o(x,y,z,t), & with\ probability\ 1-p \\ N(x,y,z,t), & with\ probability\ p \end{cases} \quad (1)$$

Where

p is the probability between [0,1] with which a pixel component value is noisy,

f_n is the noisy sequence,

f_o is the original (noisefree) sequence,

N(x, y, z, t) is the noise value of the pixel,

f_o(x, y, z, t) is the original noisefree value of the pixel.

(x, y) corresponds to the pixel location, z corresponds to one of the three color component and t corresponds to the frame number. For e.g. f(x, y, 1, t) corresponds to the red component of pixel at x^{th} row and y^{th} column in the current frame t. z=1, 2, 3 for Red, Blue and Green components respectively. This notation is followed throughout the thesis.

A color video comprises of subsequent colored frames. A colored frame can be represented in one of the several color models such as RGB, CMY, YUV, CMYK, HSV, HLS and CIE. Each industry that uses color employs the most suitable color model. For e.g. YUV or YCbCr are used in video systems, RGB is used in computer graphics etc. In the current work, RGB color

model is used to represent the frames of the color video. The importance of this color model is that it relates very closely to the way that the human eye perceives color.

3.2 Filtering steps

The impulse noise from the video frames is removed in 2 successive steps of filtering. The first step detects the noisy pixels along with the amount of noise present and are then corrected using the median of the noise-free pixels. The output of the first step acts as an input to the second filtering step and further refines the result to give the final output. The degree of noise is calculated by using a fuzzy rule which requires calculation of six fuzzy membership functions.

Most of the filters developed for video are 2D filters that filter each of the frames of the video successively. They consider only one frame at a time and remove the noise. However by considering only one frame at a time they neglect the temporal relation between frames of the video. The filter presented in this thesis is a 3D filter that considers spatial, temporal as well as the color information. A pixel in one color component is compared with its neighboring pixels within the same frame, with the corresponding pixels in neighboring frames and also with the pixels in other 2 color components. Comparison is done by using fuzzy membership functions and then applying a fuzzy rule which decides whether the given pixel is noisy or noisefree. The six membership functions are used in this work are- Big, Dissimilar, Terminal, High, Variant and Temporal. Big, Dissimilar and Terminal membership function give spatial information about a given pixel. High, Variant and Temporal membership function give temporal information about the pixel.

3.2.1 First Filtering Step

The primary objective of first step is detection and correction of noisy pixels. The noisy pixels are detected in this step along with the amount with which the pixels are corrupted. After

detection, the noisy pixels are corrected by using the median of the noisefree pixels present in the neighborhood.

Fuzzy Rule

Consider one color component, say red component of the current frame t in noisy video. The degree of noise present in a pixel at location (x, y) is achieved by the following fuzzy rule:

Rule 1:

IF ((| f (x,y,1,t) - med (x,y,1,t) | is ***Big)*** AND (the degree of similarity of this central pixel f(x,y,1,t) to its neighboring pixel in the same frame t is ***Dissimilar)*** AND (the central pixel is ***Terminal)*** AND ((either the difference between the pixel f(x,y,1,t) in the current frame t and the corresponding pixel f (x,y,1,t-1) in the previous frame t-1 is ***High)*** OR (the difference between the pixel f(x,y,1,t) in the current frame and the corresponding pixel f(x,y,1,t+1) in the next frame t+1 is ***High))*** AND (the degree of similarity of this central pixel f(x,y,1,t) to its neighboring pixels in the previous frame t-1 is ***Variant)*** AND ((|f(x,y,1,t)-f(x,y,1,t-1)| is big positive) AND (either *|f(x,y,2,t)-f(x,y,2,t-1)| is big positive) OR (|f(x,y,3,t)-f(x,y,3,t-1)| is big positive)))* THEN this pixel f(x,y,1,t) is noisy.

This rule is of the Mamdani fuzzy model type. Mathematically the degree of noise present in the red component of a pixel at location (x, y) in the current frame t is evaluated as:

$$N_1(x,y,1,t) = \min\{ \mu_b(D(x,y,1,t)), \mu_d(K_r, D(x,y,1,t)), \mu_{ter}(x, y, 1, t), \mu_h(x, y, 1, t), \mu_v(x,y,1,t), \mu_{tem}(x, y, 1, t) \} \quad (2)$$

Where

$\mu_b, \mu_d, \mu_{ter}, \mu_h, \mu_v,$ *and* μ_{tem} *are* six membership functions that represent the six fuzzy sets Big, Dissimilar, Terminal, High, Variant and Temporal respectively.

D is the difference between the central pixel and the median of the noise-free pixels in the neighborhood of a window. Mathematically,

$$D(x, y, 1, t) = |f(x, y, 1, t) - med(x, y, 1, t)| \quad (3)$$

Similarly the difference for other 2 color components is calculated by replacing 1 in the above equation by 2 and 3. This is followed throughout the thesis.

Algorithm to compute the median []:

However not all the pixels in the neighborhood of the pixel under interest are taken to compute the median, only the noise-free pixels are taken and their median is computed to modify the noisy pixel under interest. Median is computed separately for each color component. For obtaining the median of noise free pixels, an algorithm proposed in [3] is used. The Algorithm is as follows:

Step 1) Take a window of size 3X 3 centered on the currently processed pixel of the noisy frame.

Step 2) Arrange the pixels of the window as a vector. Sort the vector in the increasing order and obtain the median of the sorted vector.

Step 3) Compute the difference between each window pixel and the median calculated above.

Step 4) Arrange all the window pixels having the differences less than or equal to a parameter δ_1 into a vector.

Step 5) Sort the new vector and obtain the median (med) of the sorted vector.

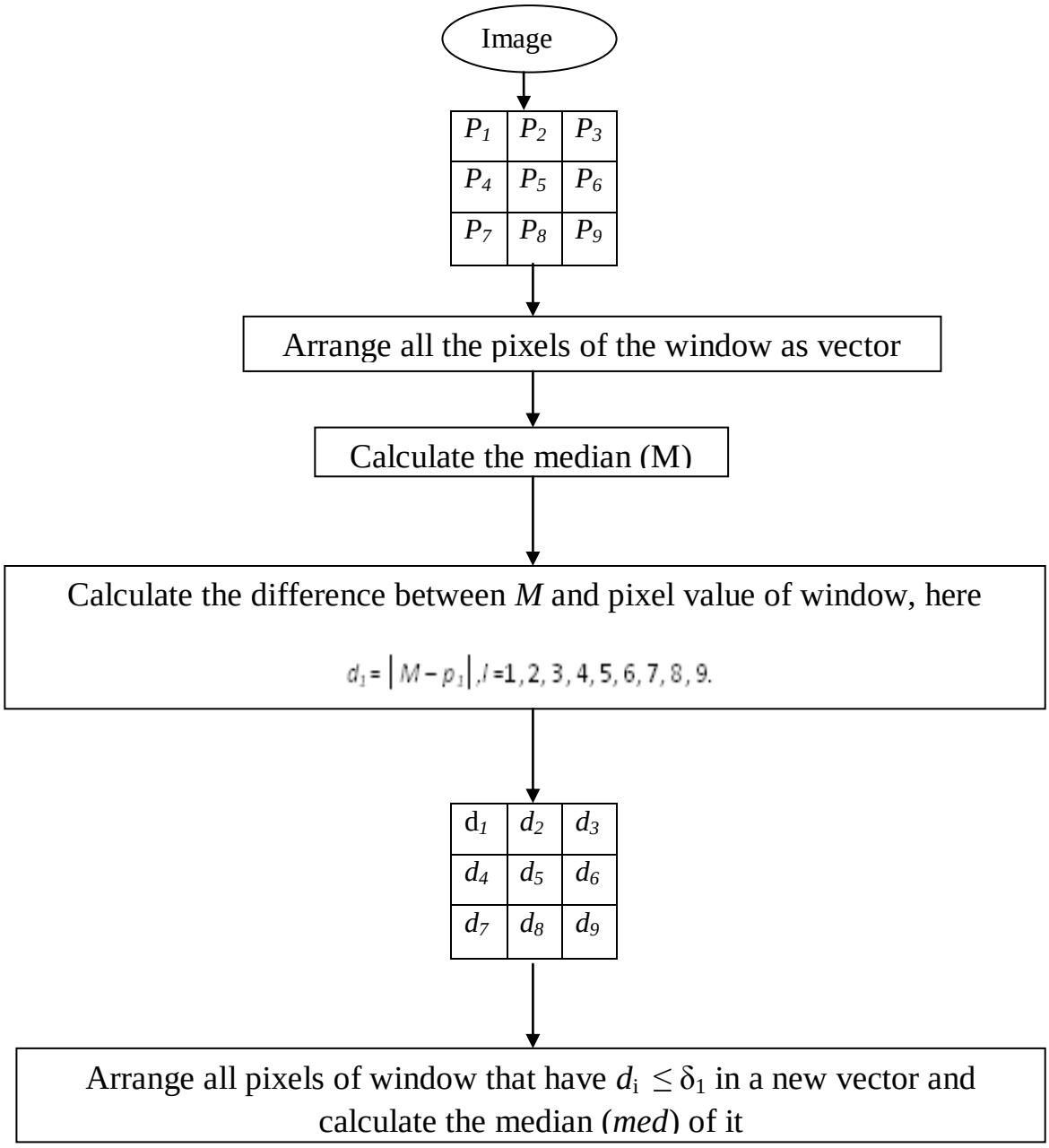


Figure 3.1 A scheme for computation of Median of the noise-free pixels [3]

A. Big fuzzy set

A pixel with more noise will have bigger difference with the median value. This is defined by the membership function μ_b μ_b. μ_b is used to represent a fuzzy set "**Big"**, that indicates how big the difference is. After calculating the difference *D*, μ_b is calculated for each color component as:

$$\mu_b(D(x,y,1,t)) = \begin{cases} 1, & D(x,y,1,t) \geq \alpha_2 \\ \dfrac{D(x,y,1,t)-\alpha_1}{\alpha_2-\alpha_1}, & \alpha_1 \leq D(x,y,1,t) < \alpha_2 \\ 0, & D(x,y,1,t) < \alpha_1 \end{cases} \quad (4)$$

The parameters α_1 and α_2 in the above equation is obtained by applying bacteria foraging. The membership function μ_b of the fuzzy set "Big" is depicted in figure 3.2 below.

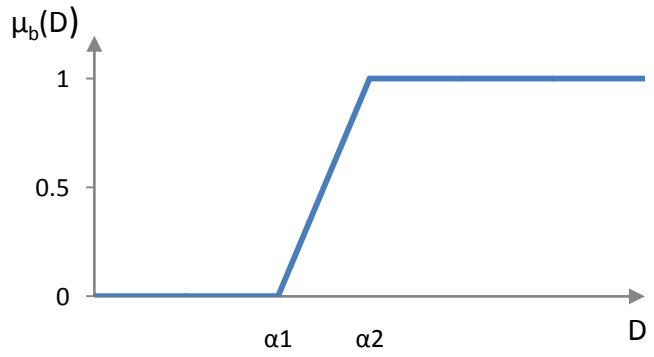


Figure 3.2: The membership function μ_b of the fuzzy set *Big*.

B. Dissimilar fuzzy set

K in equation (2) is the number of neighborhood pixels in the current frame that are similar to the given pixel. The more the number of similar pixels, the less the chances of given pixel being noisy. For this, a similarity criterion is used to decide whether the given pixel is similar to its neighborhood pixels or not. Also if we want to find the similarity of a red component of a pixel,

we will find its interaction with other 2 color components as well to ascertain the extent of similarity.

For the red component

To find the similarity of the red component of a pixel at (x, y) with its neighboring pixels at (x+i, y+j), we have to find the following differences.

The differences between red and green component and that between red and blue components are computed as follows:

$$d_{rg}(x, y, t) = \left|f(x, y, 1, t) - f(x, y, 2, t)\right| \quad (5)$$

$$d_{rb}(x, y, t) = \left|f(x, y, 1, t) - f(x, y, 3, t)\right| \quad (6)$$

Similarly the differences between red and green components and that between red and blue components of the neighboring pixels (x+i, y+j) are calculated as:

$$d_{rg}(x+i, y+j, t) = \left|f(x+i, y+j, 1, t) - f(x+i, y+j, 2, t)\right| \quad (7)$$

$$d_{rb}(x+i, y+j, t) = \left|f(x+i, y+j, 1, t) - f(x+i, y+j, 3, t)\right| \quad (8)$$

The second differences of the above pair-wise differences are computed from the following equation:

$$\Delta_{rg}(x+i, y+j, t) = \left|d_{rg}(x+i, y+j, t) - d_{rg}(x, y, t)\right| \quad (9)$$

$$\Delta_{rb}(x+i, y+j, t) = \left|d_{rb}(x+i, y+j, t) - d_{rb}(x, y, t)\right| \quad (10)$$

We also need the differences between the neighboring pixels and the central pixel of the same color component in the window given by**:**

$$\Delta_{r}(x+i, y+j, t) = \left|f(x+i, y+j, 1, t) - f(x, y, 1, t)\right| \quad (11)$$

Now the red component of a pixel at location (x,y) in the current frame is considered similar to that at (x+i, y+j) in the same current frame t if the differences $\Delta_{rg}(x+i, y+j, t)$, $\Delta_{rb}(x+i, y+j, t)$ and $\Delta_r(x+i, y+j, t)$ as calculated in equation (9), (10) and (11) respectively are less than the parameter δ_2. This parameter δ_2 is again obtained by applying bacteria foraging.

Consider a window of size w × w (say 3×3). Then i and j in equation (7), (8), (9), (10) and (11) will range from -1 to +1. There will be total 9 pixels in the window including the central pixel. K is the number of similar pixels (excluding the central pixel) in the window. That is, K is the number of neighborhood pixels that are similar to the central pixel. In the window of size 3×3, K will range from 0 to 8. More are the number of similar pixels in the neighborhood; less will be the chances of central pixel being noisy. This number K is decided by comparing differences calculated in Equations (9), (10) and (11) with parameter δ_2.

Now if $\Delta_{rg}(x+i, y+j, t)$, $\Delta_{rb}(x+i, y+j, t)$ and $\Delta_r(x+i, y+j, t)$ are each less than a parameter δ_2 we say that condition A meets, then

$$K_r(x,y,t) = \begin{cases} K_r(x,y,t)+1, & condition\ A\ meets \\ K_r(x,y,t), & otherwise \end{cases} \qquad (12)$$

A pixel with high number of similar neighborhood pixels and whose value is close to the median will be more likely to be noisefree. This is defined by the membership function μ_d. μ_d is used to represent a fuzzy set "**dissimilar"**, that indicates the degree of similarity of the central pixel to its neighborhood pixels. After calculating the difference Dif and the number of similar pixels for the central pixel in a window of size w × w (taking 3×3 here), μ_d is calculated for each color component as:

$$\mu_d\left(K_r,\ D(x,\ y,\ 1,\ t)\right) = \max\left(\mu\left(K_r(x,\ y,\ t)\right),\ \mu\left(D(x,\ y,\ 1,\ t)\right)\right) \qquad (13)$$

Where membership function $\mu(K_r)$ and $\mu(D(x, y, 1, t))$ are depicted in figures 3.3 and 3.4 below respectively.

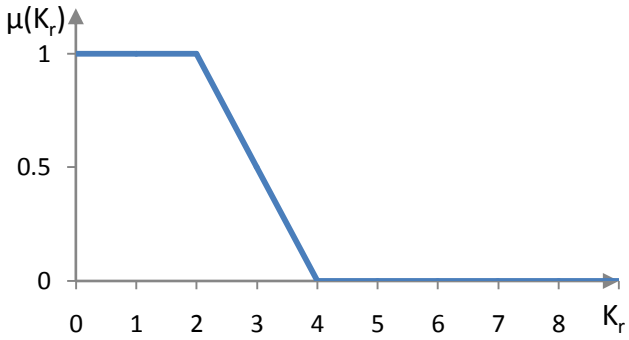


Figure 3.3: The membership function $\mu(K_r)$

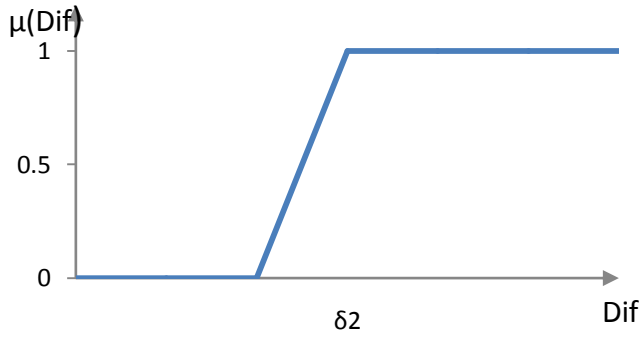


Figure 3.4: The membership function μ(D)

Here D(x, y, 1, t) is same as that calculated in equation (3) and max operator used is the Cartesian co-product. Therefore if a pixel has more than half pixels similar in the window and its value is close to the median, then it can be considered as a noise-free pixel.

C. Terminal fuzzy set

Now Let us take a window of size w× w (say 3×3). There are 9 pixels in this window, if we arrange these nine pixels in increasing order of their intensity; we will get 2 terminal pixels and one median pixel. The 2 terminal pixels are denoted by f_{min} having the smallest value in the window and f_{max} having the highest value in the window respectively and median pixel is denoted by f_{med}. The closer the value of central pixel to these terminal values f_{min} and f_{max}, the higher its probability of being noisy and the closer the value of the pixel to f_{med}, the higher its probability of being noisefree. This is defined by the third membership function μ_{ter}. μ_{ter} is used to represent a fuzzy set "**terminal"**. μ_{ter} is calculated for each color component as:

$$\mu_{ter}(\omega)=\begin{cases} 1, & \omega = f_{\min} \; or \; \omega = f_{\max} \\ \dfrac{\omega - f_{\min}}{f_{med} - f_{\min}}, & f_{\min} < \omega < f_{med} \\ \dfrac{f_{\max} - \omega}{f_{\max} - f_{med}}, & f_{med} < \omega < f_{\max} \\ 0, & \omega = f_{med} \end{cases} \qquad (14)$$

Where ω represents the pixel value of given color component of the given frame.

The membership function for the fuzzy set **Terminal** is shown below in figure 3.5:

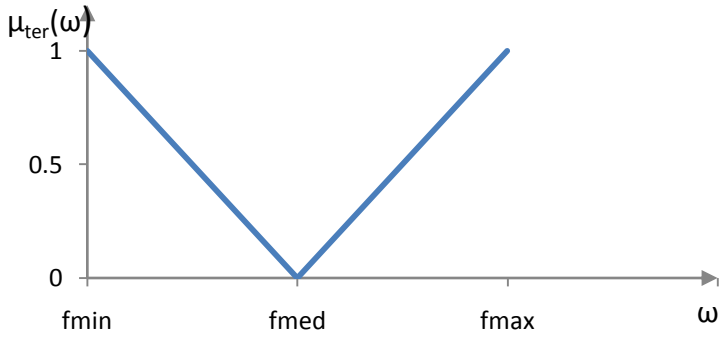


Figure 3.5: The membership function $\mu_{ter}(\omega)$ for the fuzzy set *terminal*

Till now we have only focused on the color and spatial information. But since we are removing noise from a color video, we also need to take care of the temporal information. If we want to calculate the degree of noise present in a component of a pixel in one frame, we need to

consider its neighboring frames (previous and next frame) too. Therefore the next three membership functions $\mu_{h,}\ \mu_{v,}$ *and* μ_{tem} are used to calculate the degree of noise present in a component of the pixel in one frame by comparing the current frame with its neighboring frames.

D. High fuzzy set

A noisy pixel component in one frame will not only have high difference with its corresponding component neighboring pixels in the same frame but also with corresponding component of the pixel at the same spatial location in the previous or next frame. This concept is used in obtaining the fuzzy set "**high**" defined by the membership function μ_h. That is, if a component (let us say Red) of a pixel at (x,y) in the current frame (t) has a high positive difference with either the corresponding component(Red) of a pixel at the same spatial location (x,y) in the previous frame (t-1) or with the corresponding component(Red) of a pixel at the same spatial location (x,y) in the next frame (t+1),it will be considered as High. The parameters γ_1 and γ_2 are again obtained by applying bacteria foraging.

Let,

$$R_1 = \left| f(x, y, 1, t) - f(x, y, 1, t-1) \right|$$

$$R_2 = \left| f(x, y, 1, t) - f(x, y, 1, t+1) \right|$$

Then μ_h is calculated for red color component as:

$$\mu_h(x, y, 1, t) = \max\left(\mu\left(R_1(x, y, 1, t)\right), \mu\left(R_2(x, y, 1, t)\right)\right) \qquad (15)$$

Where membership function $\mu\,(R_1(x, y, 1, t))$, and $\mu\,(R_2(x, y, 1, t))$ are depicted in figures 3.6 and 3.7 below respectively and max operator used is the Cartesian co-product.

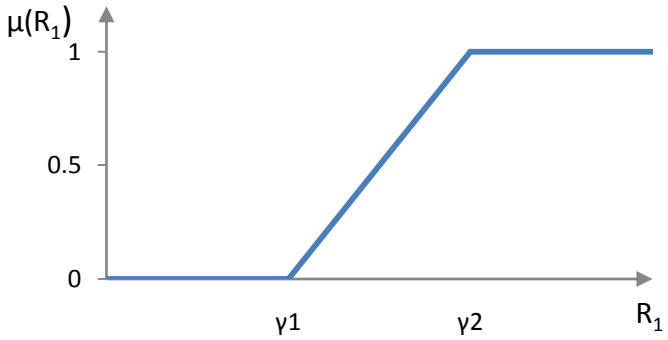


Figure 3.6: The membership function $\mu(R_1)$

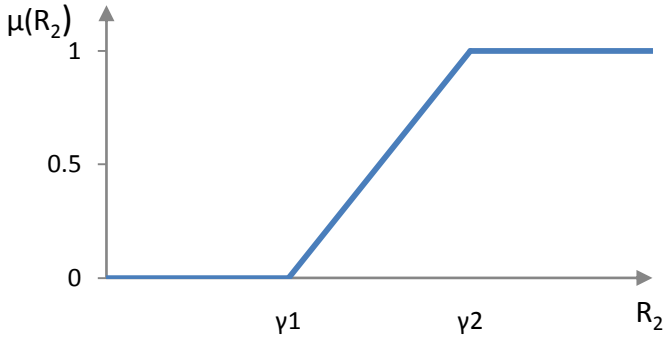


Figure 3.7: The membership function $\mu(R_2)$

E. Variant fuzzy set

For the red component

To find the similarity of the red component of a pixel at (x, y) in the current frame (t) with corresponding component of the pixel at the same spatial location (x, y) in the previous frame (t-1), we have to find the following differences.

The difference between a pixel component in the current frame (t) and the corresponding component of the pixel at the same spatial location in the previous frame (t-1) are computed as follows:

$$R_1(x, y, t) = \left|f(x, y, 1, t) - f(x, y, 1, t-1)\right| \quad (16)$$

$$G_1(x, y, t) = \left|f(x, y, 2, t) - f(x, y, 2, t-1)\right| \quad (17)$$

$$B_1(x, y, t) = \left|f(x, y, 3, t) - f(x, y, 3, t-1)\right| \quad (18)$$

Similarly the differences between neighboring pixels (x+i, y+j) of the pixel component in the current frame (t) and that between the neighboring pixels (x+i, y+j) of the corresponding component of the pixel at the same spatial location in the previous frame (t-1) are calculated as:

$$d_r(x+i, y+j, t) = \left|f(x+i, y+j, 1, t) - f(x+i, y+j, 1, t-1)\right| \quad (19)$$

$$d_g(x+i, y+j, t) = \left|f(x+i, y+j, 2, t) - f(x+i, y+j, 2, t-1)\right| \quad (20)$$

$$d_b(x+i, y+j, t) = \left|f(x+i, y+j, 3, t) - f(x+i, y+j, 3, t-1)\right| \quad (21)$$

The second differences of the above pair-wise differences are computed from the following equation:

$$R_3(x+i, y+j, t) = \left|d_r(x+i, y+j, t) - R_1(x, y, t)\right| \quad (22)$$

$$G_3(x+i, y+j, t) = \left|d_g(x+i, y+j, t) - G_1(x, y, t)\right| \quad (23)$$

$$B_3(x+i, y+j, t) = \left|d_b(x+i, y+j, t) - B_1(x, y, t)\right| \quad (24)$$

If the given pixel component at (x, y) in the current frame is similar to the corresponding pixel component at the same spatial location(x, y) in the previous frame, then the neighborhood pixels

(x+i, y+j) of the pixel component in the current frame will also be similar to the neighboring pixels (x+i, y+j) of the corresponding component of the pixel at the same spatial location in the previous frame. We take a window of size w × w (say 3×3) and find such number of neighborhood pixels around the given pixel which are similar to the neighboring pixels of the corresponding component of the pixel at the same spatial location in the previous frame. We call this number as k as used in equation number (2). That is, k is the number of neighborhood pixels (x+i, y+j, t) in the current frame that are similar to corresponding neighborhood pixels (x+i, y+j, t-1) in the previous frame for the same color component. In the window of size 3×3, k will range from 0 to 8. This number k is calculated for each color component separately by using the parameter δ_2. This parameter δ_2 is same as that used before. For the Red component,

$$k_r(x,y,t) = \begin{cases} k_r(x,y,t)+1, & R_3(x+i,y+j) < \delta_2 \\ k_r(x,y,t), & otherwise \end{cases} \qquad (25)$$

More are the number of similar neighborhood pixels; more are the chances of central pixel in current frame being similar to the corresponding pixel in the previous frame and hence less will be the chances of central pixel being noisy. A pixel with high number of such similar neighborhood pixels for the same color component and for at least one of the other 2 component can be considered as noisefree. This is defined by the membership function μ_v. μ_v is used to represent a fuzzy set "**variant"**, that indicates the degree of similarity of the central pixel in the current frame to its corresponding pixel at the same spatial location in the previous frame. After calculating the number of similar pixels for the central pixel in a window of size w × w (taking 3×3 here), μ_v is calculated for red color component as:

$$\mu_v(x,\ y,\ 1,\ t)\ =\ min\ \left(\mu(k_r),\ max\left(\mu(k_g),\ \mu(k_b)\right)\right) \qquad (26)$$

For simplicity we have used k_r for k_r(x, y, 1, t), k_g for k_g(x, y, 1, t) and k_b for k_b(x, y, 1, t) .

These membership function μ (k_r(x, y, 1, t)), μ (k_g(x, y, 1, t)) and μ (k_b(x, y, 1, t)) are depicted in figures 3.8, 3.9 and 3.10 below respectively. Max operator used is the Cartesian co-product and min operator used is the Cartesian product.

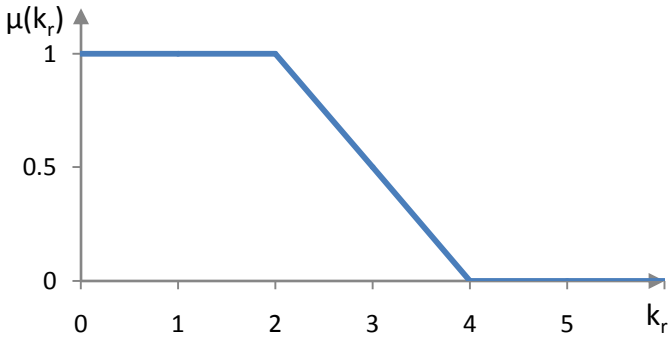


Figure 3.8: The membership function $\mu(k_r)$

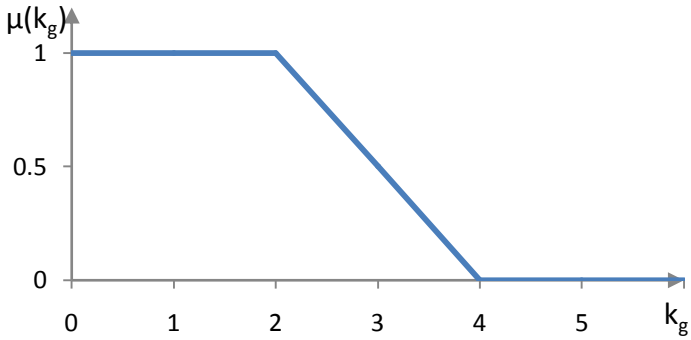


Figure 3.9: The membership function $\mu(k_g)$

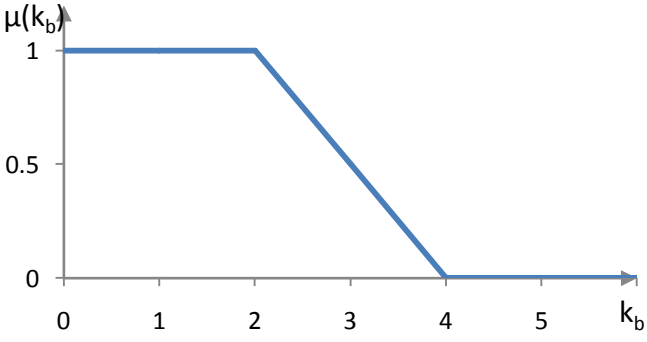


Figure 3.10: The membership function $\mu(k_b)$

F. Temporal fuzzy set

The sixth membership function μ_{tem} in equation number (2) represents a fuzzy set "**Temporal"** that takes into account the temporal as well as the color information at a time to find the degree of noise present in the pixel. As we know from equation number (16), (17) and (18), the difference between a pixel component in the current frame (t) and the corresponding component of the pixel at the same spatial location in the previous frame (t-1) are computed as follows:

$$R_1(x,\ y,\ t) \ = \ \left|f(x,\ y,\ 1,\ t) - f(x,\ y,\ 1,\ t-1)\right|$$

$$G_1(x,\ y,\ t) \ = \ \left|f(x,\ y,\ 2,\ t) - f(x,\ y,\ 2,\ t-1)\right|$$

$$B_1(x,\ y,\ t) \ = \ \left|f(x,\ y,\ 3,\ t) - f(x,\ y,\ 3,\ t-1)\right|$$

For the red component, μ_{tem} is calculated as,

$$\mu_{tem}(x,\ y,\ 1,\ t) \ = \ min\ \left(\mu(R_1),\ max\left(\mu(G_1),\ \mu(B_1)\right)\right) \qquad (27)$$

Again for simplicity we have used R_1 for R_1 (x, y, 1, t), G_1 for G_1 (x, y, 1, t) and B_1 for B_1 (x, y, 1, t).

These membership function μ (R_1 (x, y, 1, t)), μ (G_1 (x, y, 1, t)) and μ (B_1 (x, y, 1, t)) are depicted in figures 3.11, 3.12 and 3.13 below respectively. Max operator used is the Cartesian co-product and min operator used is the Cartesian product.

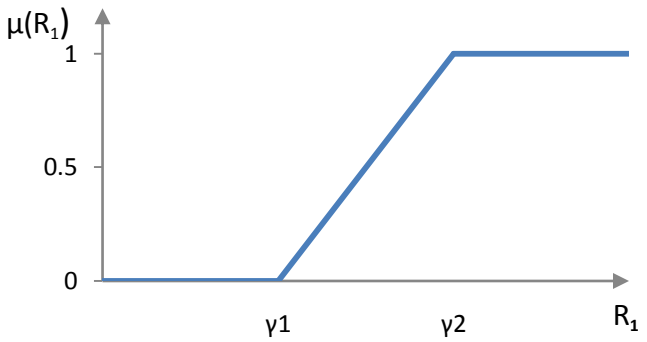


Figure 3.11: The membership function $\mu(R_1)$

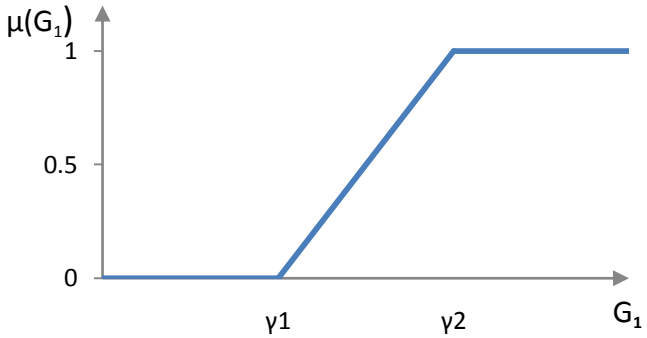


Figure 3.12: The membership function $\mu(G_1)$

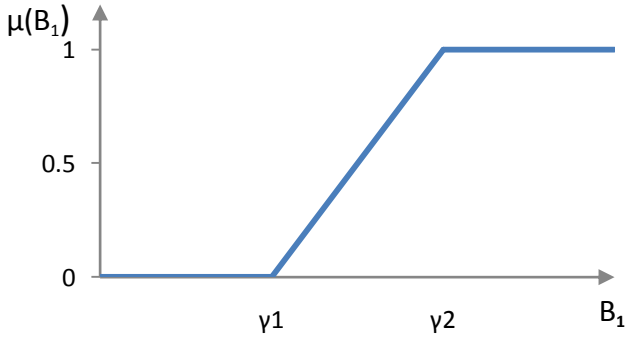


Figure 3.13: The membership function $\mu(B_1)$

Thus, using equation number (4), (13), (14), (15), (26) and (27) and applying them in equation number (2), we will get $N_1(x,y,1,t)$, that is :

$N_1(x,y,1,t) = min\{ \mu_b(D(x,y,1,t)), \mu_d(K_r, D(x,y,1,t), \mu_{ter}(x, y, 1, t), \mu_h(x, y, 1, t), \mu_v(x,y,1,t), \mu_{tem}(x, y, 1, t) \}$

Since the fuzzy rule is of the Mamdani fuzzy model type, the AND operator is used to implement the "*minimum"* operation. As we know $N_1(x, y, 1, t)$ is the degree of noise present in the red component of a pixel at location (x, y) in the current frame (t). Its value should be zero for the noisefree pixel and for the noisy pixel; it should have some value between 0 and 1.

Correction Term

Correction term is a term which is added to the original value of the pixel to make it noise free. So if the pixel is already noisefree the correction term will be zero otherwise for noisy pixel, correction term will have some positive or negative value. The correction term for the red component is computed as:

$$\Delta f(x,y,1,t) = N_1(x,y,1,t) \times \left(med(x,y,1,t) - f(x,y,1,t) \right) \quad (28)$$

Thus if a pixel is noise free the degree of noise present in it will be zero and hence the correction term will be zero and for extremely noisy pixel(N_1=1), the value of correction term is equal to the difference between the median (med) of noise free pixels in the neighborhood and the value of the pixel itself.

This correction term obtained is then added to the original value of the pixel. The uncorrupted pixels are left untouched and highly corrupted pixels are replaced by the median value of the noisefree pixels in their neighborhood. Other pixels having average noise density are modified according to the degree of noise present in them. Thus the result of first filtering step is:

$$f_1(x,y,1,t) = f(x,y,1,t) + \Delta f(x,y,1,t) \quad (29)$$

3.2.2 Second Filtering Step

Now the result of the first filtering step acts as an input to the second filtering step to give the final output. This step refines the output of first filtering step by taking into account the interactions among the color components of the same frame as well as the interaction of the given frame with its previous frame as well as next frame.

The objective of second filtering step is detection and correction of residual noisy pixels. The noisy pixels are detected along with the amount with which they are corrupted. After detection, the noisy pixels are corrected by using the median of the noisefree pixels present in the neighborhood.

Again in the second filtering step, the degree of noise is calculated by using a fuzzy rule. Consider one color component, say red component of the current frame t in the resulting video of the first step f_1. The degree of noise present in a pixel at location (x, y) is achieved by the following fuzzy rule:

Fuzzy Rule 2:

IF ($| f_1(x,y,1,t) - f_1(x,y,2,t) |$ is ***Big)*** AND ($| f_1(x,y,1,t) - f_1(x,y,3,t) |$ is ***Big)*** AND ((either the difference between the pixel $f_1(x,y,1,t)$ in the current frame t and the corresponding pixel $f_1(x,y,1,t-1)$ in the previous frame t-1 is ***High)*** OR (the difference between the pixel $f_1(x,y,1,t)$ in the current frame and the corresponding pixel $f_1(x,y,1,t+1)$ in the next frame t+1 is ***High))*** THEN this pixel $f_1(x,y,1,t)$ is noisy.

This rule is of the Mamdani fuzzy model type. Mathematically the degree of noise present in the red component of a pixel at location (x, y) in the current frame t is evaluated as:

$$N_2(x, y, 1, t) = min\{ \mu_b(\zeta_{rg}(x, y, t)), \mu_b(\zeta_{rb}(x, y, t)), \mu_h(x, y, 1, t)\} \qquad (30)$$

Where

μ_b *and* μ_h *are* the same membership functions used above that represent the fuzzy sets "Big" and "High" respectively.

ζ_{rg} *is t*he difference between red and green component and ζ_{rb} is the difference between red and blue components and are computed as follows:

$$\zeta_{rg}(x, y, t) = \left| f_1(x, y, 1, t) - f_1(x, y, 2, t) \right| \tag{31}$$

$$\zeta_{rg}(x, y, t) = \left| f_1(x, y, 1, t) - f_1(x, y, 3, t) \right| \tag{32}$$

After calculating these differences, we will compute fuzzy membership value μ_b by comparing these differenc values of these 2 parameters are obtained through bacteria foraging.

$$\mu_b(\zeta_{rg}(x, y, t)) = \begin{cases} 1, & \zeta_{rg}(x, y, t) \geq \beta_2 \\ \dfrac{\zeta_{rg}(x, y, t) - \beta_1}{\beta_2 - \beta_1}, & \beta_1 \leq \zeta_{rg}(x, y, t) < \beta_2 \\ 0, & \zeta_{rg}(x, y, t) < \beta_1 \end{cases} \tag{33}$$

$$\mu_b(\zeta_{rb}(x, y, t)) = \begin{cases} 1, & \zeta_{rb}(x, y, t) \geq \beta_2 \\ \dfrac{\zeta_{rb}(x, y, t) - \beta_1}{\beta_2 - \beta_1}, & \beta_1 \leq \zeta_{rb}(x, y, t) < \beta_2 \\ 0, & \zeta_{rb}(x, y, t) < \beta_1 \end{cases} \tag{34}$$

The membership functions $\mu_b(\zeta_{rg})$ and $\mu_b(\zeta_{rb})$of the fuzzy set "Big" is depicted in figure 3.14 and 3.15 below respectively.

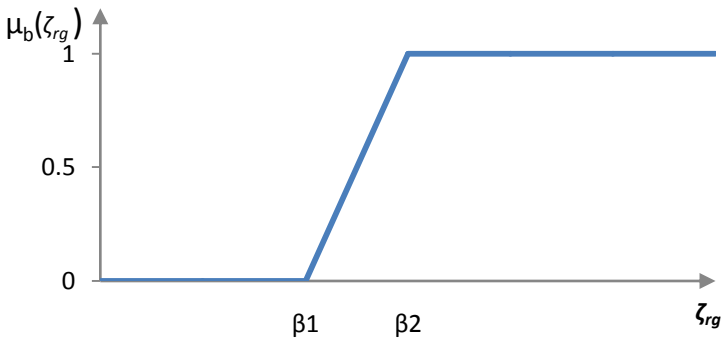


Figure 3.14: The membership function $\mu_b(\zeta_{rg})$

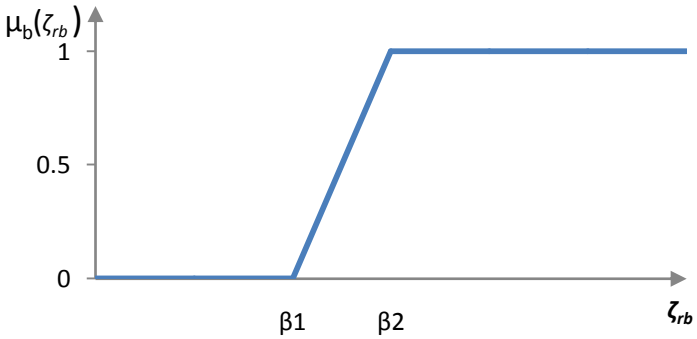


Figure 3.15: The membership function $\mu_b(\zeta_{rb})$

Now Let,

$$R_{11} = \left| f_1(x, y, 1, t) - f_1(x, y, 1, t-1) \right|$$

$$R_{21} = \left| f_1(x, y, 1, t) - f_1(x, y, 1, t+1) \right|$$

Then μ_h is calculated for red color component as:

$$\mu_h(x, y, 1, t) = \max\left(\mu\left(R_{11}(x, y, 1, t)\right), \mu\left(R_{21}(x, y, 1, t)\right)\right) \quad (35)$$

Where membership function μ (R_{11}(x, y, 1, t)), and μ (R_{21}(x, y, 1, t)) are depicted in figures 3.16 and 3.17 below respectively and max operator used is the Cartesian co-product.

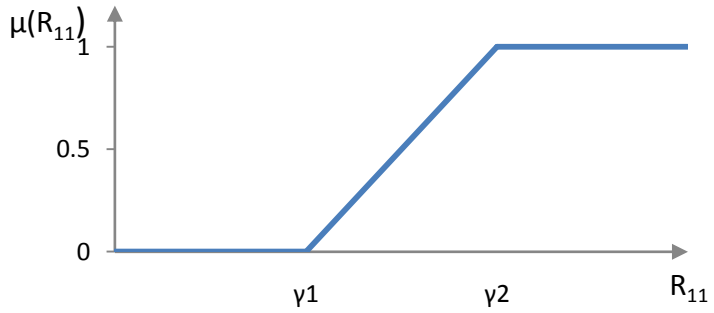


Figure 3.16: The membership function μ(R_{11})

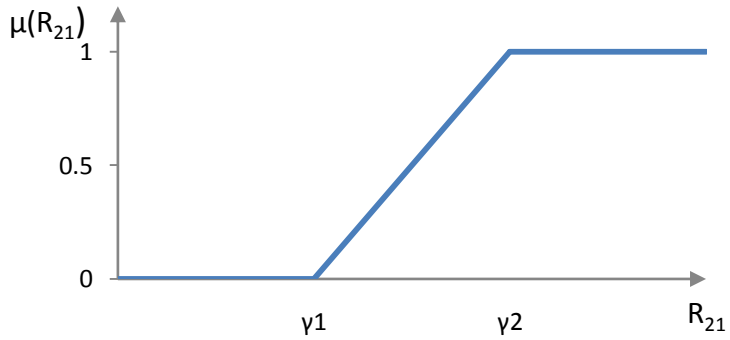


Figure 3.17: The membership function μ(R_{21})

Again the fuzzy rule is of the Mamdani fuzzy model type, the AND operator is used to implement the "*minimum"* operation. As we know $N_1(x, y, 1, t)$ is the degree of noise present in the red component of a pixel at location (x, y) in the current frame (t). Its value should be zero for the noisefree pixel and for the noisy pixel; it should have some value between 0 and 1.

The correction term for the red component is computed as:

$$\Delta f_2(x,y,1,t) = N_1(x,y,1,t) \times \left(\text{med}(x,y,1,t) - f_1(x,y,1,t)\right) \quad (36)$$

This correction term obtained is then added to the original value of the pixel. The uncorrupted pixels are left untouched and highly corrupted pixels are replaced by the median value of the noisefree pixels in their neighborhood. Other pixels having average noise density are modified according to the degree of noise present in them. Thus the result of second filtering step is:

$$f_2(x, y, 1, t) = f(x, y, 1, t) + \Delta f(x, y, 1, t) \quad (37)$$

CHAPTER 4

OPTIMIZATION USING BACTERIA FORAGING

4.1 Evolutionary Algorithms

Optimization techniques may follow different approaches. Traditional analytical optimization techniques require enormous computational efforts, which grow exponentially as the problem size increases. An optimization method that requires moderate memory and computational resources and yet produces good results is desirable. Recently, optimization techniques inspired by biology behaviours, known as bio-mimetic optimization algorithms, have obtained more and more attention [44]. Bio-inspired optimization methods are computationally efficient alternatives to analytical methods. Evolutionary Algorithms such as Genetic Algorithms (GAs) [45-50], Particle Swarm Optimization (PSO) [51-53], and Ant Colony Optimization (ACO) [54-58], are some popular multidimensional optimization techniques. Bio-mimetic optimization algorithms are developed from simulation the evolutionary process and the behaviours of biology. They are population-based (each member stands for a biology individual), and initialized with a population of individuals. They utilize the direct information "fitness" instead of individual's ability to adapt to the environment. These individuals are manipulated over many generations by ways of mimicking social behaviour of biology, in an effort to find the optima. The advantage of evolutionary algorithms compared to other optimization methods is their "black box" character that makes only few assumptions about the underlying objective functions. Furthermore, the definition of objective functions usually requires lesser insight to the structure of the problem space than the manual construction of an admissible heuristic. EAs therefore perform consistently well in many different problem categories. In comparison with other optimization algorithms, bio-mimetic optimization algorithms have the following characteristics:

1) The individual components are distributed and autonomous, there is no central control, and the fault of an individual cannot influence solving the whole problem, these characteristics ensure this kind of algorithms has better robustness.

2) The manner of achieves individual collaboration though not directly information communication make sure of the expansibility of the algorithm.

3) They don't demand to meet the requirement of differentiability, convexity and other conditions for mathematical description of the problem.

4) Because of concerns merely with basic mathematical operations, therefore, they are simple and easy to be implemented on computer.

Natural selection tends to eliminate animals with poor foraging strategies and favour the propagation of genes of those animals that have successful foraging strategies, since they are more likely to enjoy reproductive success. After many generations, poor foraging strategies are either eliminated or shaped into good ones. This activity of foraging led the researchers to use it as optimization process. Based on the researches on the foraging behaviour of E. coli bacteria, Prof. K. M. Passino proposed a new evolutionary computation technique known as Bacterial Foraging Optimization Algorithm (BFOA) [59, 60]. In BFOA, the foraging (methods for locating, handling, and ingesting food) behaviour of E. coli bacteria is mimicked.

4.2 Brief overview of Classical Bacteria Foraging Optimization Algorithm

Bacteria Foraging is a bacterial-derivative based optimization algorithm proposed by passino [59]. BFOA [61-63] exploits the foraging behaviour of bacteria to solve the real-world optimization problems. The BFO is a non-gradient optimization problem inspired by the foraging strategy of the E.Coli bacteria. Foraging is the method of locating, handling and ingesting the food. Bacteria seek to minimize the effort spent per unit time in foraging or we can say bacteria seek to maximize the energy obtained per unit time spent during foraging. An objective function is the cost function or the effort spent by the bacteria during foraging. In the beginning, there will be as many solutions as the number of bacteria. Out of many bacteria engaged in foraging (several solution paths), some come out successful in achieving an optimum cost (an optimum solution). The foraging strategy of E. coli bacteria is mainly governed by four processes: chemo taxis, swarming, reproduction, and elimination and dispersal [64]. Chemo taxis stage is the movement stage of bacteria achieved through swimming and tumbling. Swimming is the process of movement in the same direction as the previous step and Tumbling is the process of movement in an absolutely different direction from the previous one. To reach to the optimal path as quickly as possible, it is necessary for the bacterium, which has searched the high nutrient region, to signal other bacteria. The bacterium which has found such region signals other bacteria via chemical attractants to swim together. This is achieved in the

swarming stage through cell to cell signalling. Now, least healthy bacteria are those which are in low nutrient region and healthiest bacteria are those which are in high nutrient region. In the reproduction stage, the least healthy bacteria die and of the healthiest, each bacterium split into two bacteria, which are then placed in the same location. This makes the population of bacteria constant. Finally, the Elimination and dispersal stage helps the bacteria from being trapped in a premature solution point or local optima instead of global optima. There might be a possibility that bacteria have found a nutrient region which is good enough but not the best. This is the local optima. To look for new nutrient, some Bacteria from the total set get eliminated and dispersed with some probability to prevent stagnation.

Among all the evolutionary algorithms, BFA being the latest trend that is efficient in optimizing parameters of the structures. Nowadays Bacteria Foraging technique is gaining importance in the optimization problems. Because

• Philosophy says, Biology provides highly automated, robust and effective organism.

• Search strategy of bacteria is salutary (like common fish) in nature.

• Bacteria can sense, decide and act so adopts social foraging (foraging in groups).

Above all, Search and optimal foraging decision-making of animals can be used for solving engineering problems. To perform social foraging an animal needs communication capabilities and it gains advantages that can exploit essentially the sensing capabilities of the group, so that the group can gang-up on larger prey, individuals can obtain protection from predators while in a group, and in a certain sense the group can forage a type of collective intelligence.

4.3 Parameter Training For Noise Removal with Bacteria Foraging

Mean Square Error (MSE) is used as an objective function, which is optimized using the bacterial foraging algorithm to learn the parameters of membership functions. The filtering action of the proposed filter is dependent on eight parameters α_1, α_2, δ_1, δ_2, γ_1, γ_2, β_1 and β_2. In our approach we are training the values of these parameters with the help of bacterial foraging

optimization technique. The search space of bacteria foraging technique is eight dimensional (for eight parameters) and movement of bacteria find the minimum value of Mean Square Error. Parameter training with the help of bacterial foraging gives different values of parameters for different noise concentration level; this is in contrast with the given constant values for these parameters. Mean Square Error is given as

$$MSE(f,I)=\frac{\sum_{t=1}^{T}\sum_{z=1}^{3}\sum_{x=1}^{N}\sum_{y=1}^{M}\left[I(x,y,z,t)-f(x,y,z,t)\right]^2}{3\times N\times M} \tag{38}$$

Where I is the original video, f is the filtered video of N X M size. T is the total number of frames in the video. *MSE* give the similarity between two videos.

Optimization problem is to find the minima of the *MSE* so food function for bacterial foraging is defined as Mean Square Error

$$\text{Food function, } J_{minimum} = MSE \tag{39}$$

To reduce the complexity, the cell-to-cell attractant function is ignored in swarming. The selection of the initial parameters of the algorithm, such as the number of iterations and the final error value, plays a key role in deciding the accurate optimum values in lesser time. These parameters are not constant for all applications but rather depend on the application.

4.4 Initialization of Parameters

This includes two sets of parameters: the parameters in the original objective function J to be optimized, i.e., α_1, α_2, δ_1, δ_2, γ_1, γ_2, β_1 and β_2, and the parameters of the BF entering into J for facilitating the optimization. However the latter parameters are also optimized in the current work to give the best possible value for the objective function J. The initialization of the former will be discussed in the next section, and the initialization of the latter is now taken up.

1. The number of bacteria S = 10.
2. Bacteria Split Ratio S_r= S/2

3. The swimming length $N_s = 4$.
4. The number of iterations in a chemotactic loop N_c is set to 4.
5. The number of reproduction steps N_{re} is set to 1.
6. The number of elimination and dispersal events N_{ed} is set to 1.
7. The probability of elimination/dispersal P_{ed} is set to 0.25.
8. The location of each bacterium, which is a function of several parameters, i.e., f (P_{ed}, N_b, N_c, N_{re}, N_{ed}), is specified by a random number in the range [0–1].
9. Dimension of search space d = 8 (for eight parameters).

Now, the parameters of the BF entering into J for facilitating the optimization - d, N_b, N_s, N_c, N_{re}, N_{ed} and P_{ed}, are initialized as described above. After initializing these parameters, the cost (i.e., objective) function (MSE in our case) is optimized using the Bacteria Foraging algorithm and finally the optimized parameters - α_1, α_2, δ_1, δ_2, γ_1, γ_2, β_1, β_2, are calculated.

4.5 Effect on MSE after Optimization

The performance of proposed noise filter is evaluated on test colour video "salesman" with noise = 10%, 20% and 30%. The parameters $\alpha_1 = 10.5$, $\alpha_2 = 10.5$, $\delta_1 = 5$, $\delta_2 = 0.05$, $\gamma_1 = 20$, $\gamma_2 = 31$, $\beta_1 = 11$ and $\beta_2 = 11$ are given as the initial values to the Bacterial foraging algorithm. The results of the proposed approach are compared in terms of MSE before optimization and after optimization in Table 4.1 and Table 4.2 respectively.

Table 4.1 and Table 4.2 emphasises the effect of bacterial foraging optimization on the membership function parameters $\alpha_1 = 10.5$, $\alpha_2 = 10.5$, $\delta_1 = 5$, $\delta_2 = 0.05$, $\gamma_1 = 20$, $\gamma_2 = 31$, $\beta_1 = 11$ and $\beta_2 = 11$ for the noise =10%, 20% and 30%. It shows that there is a great difference between the values of the membership function parameters and MSE before and after the optimisation. The MSE for 10% noise decreases from 07.0279 to 6.8677 after optimization. This counts to 2.33% variation. However for 20% noise, it is 2.58% since the MSE for 20% noise decreases from 13.8594 to 13.5110 after optimization. For 30% noise, MSE decreases from 20.9836 to 20.2476 after optimization which means 3.63% variation.

Table 4.1: Value of MSE and membership function parameters without optimization

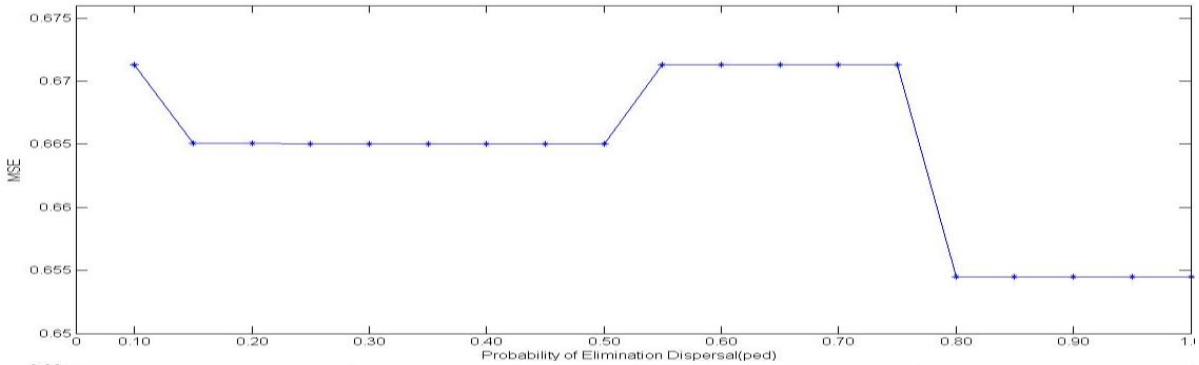
Noise (%)	Optimized Value of Parameters								MSE before Optimization
	α_1	α_2	δ_1	δ_2	γ_1	γ_2	β_1	β_2	
10	10.5	10.5	5	0.05	10.5	31	11	11	07.0279
20	10.5	10.5	5	0.05	10.5	31	11	11	13.8594
30	10.5	10.5	5	0.05	10.5	31	11	11	20.9836

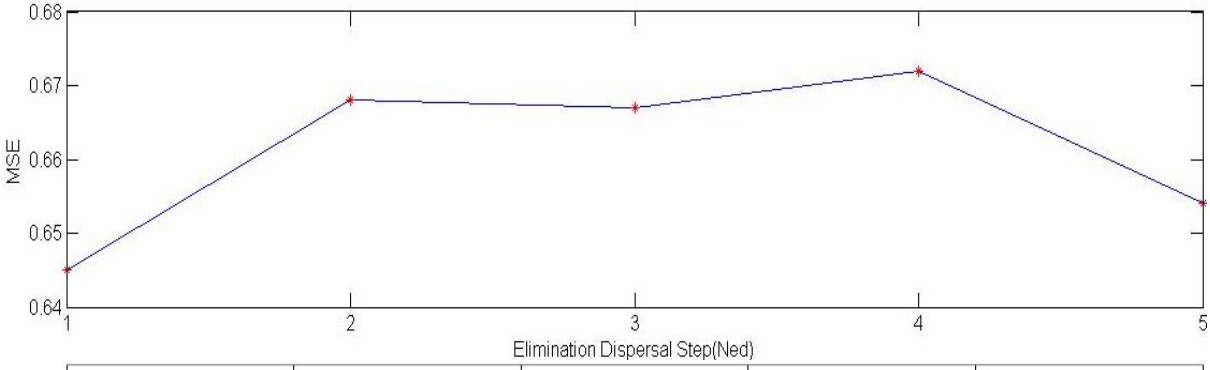
Table 4.2: Optimized Value of membership function parameters and the effect of bacterial foraging optimization on MSE

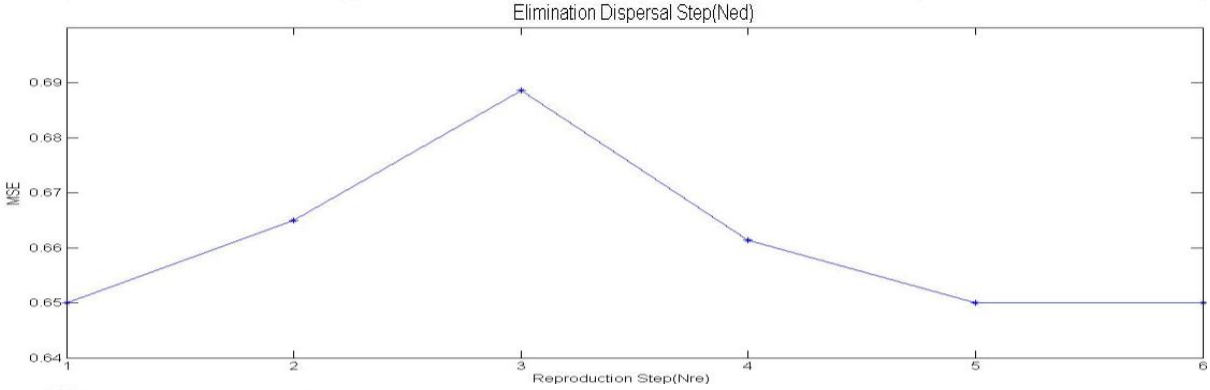
Noise (%)	Optimized Value of Parameters								MSE after Optimization
	α_1	α_2	δ_1	δ_2	γ_1	γ_2	β_1	β_2	
10	10.1177	10.2010	5.2060	0.1944	10.0065	30.0293	10.1253	10.6629	6.8677
20	9.8628	10.1108	4.9551	0.0160	10.0416	30.1606	10.3558	10.4988	13.5110
30	10.1668	10.3847	5.0183	0.0975	9.9504	30.0582	10.2088	10.7495	20.2476

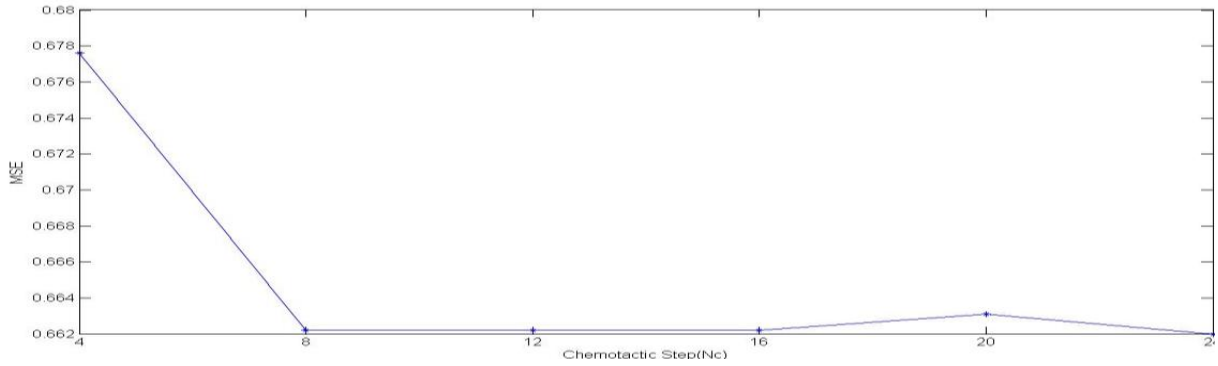
4.6 Effects of Parameter Variation

We now discuss the effects of varying different parameters of the proposed method namely: S, S_r, N_s, P_{ed}, N_{ed}, N_{re} and N_c on the video 'salesman'. The variation of parameters is judged by Mean Square Error (MSE). It is well known that an optimum value of parameter is the one with less MSE. The results for various parameters are as shown in Fig. 3.18(a)–(g). We find that N_{ed}, the elimination dispersal (fig 3.18 b) have no considerable effect on the results, MSE remains more or less the same but it is minimum at N_{ed}=1therefore we choose N_{ed} =1. Ns, the swim length (fig 3.18 e) is varied from 0 to 60 and we observe from the plot that MSE decreases first till Ns =28 after which MSE starts increasing gradually, therefore Ns =28 is found to be a suitable value for optimization. In Fig 3.18(g) we have taken different values of Sr, the Bacteria split ratio, from 0.1S to 1S, where S is the number of bacteria. We found that Sr has least MSE at 0.6S then MSE remains almost constant. Therefore Sr =0.6S is chosen to be a suitable value for optimization. Change in N_c, the chemotactic step (fig 3.18 d) from 4 to 20 causes MSE to drop steeply at NC=8 after which MSE remain almost constant, therefore N_c=8 is chosen for optimization. In Fig 3.18(f), we find that S=16 is found to be a suitable value for optimization. It may also be noted that though increase in S is favourable here but it adds more burden on computing resources. Thus, an optimal trade-off has to be found between performance and resources. In Fig 3.18(a), the probability of elimination dispersal is varied from 0.1 to 1 and we observe from the plot that MSE remain more or less the same till ped=0.8 at which MSE drops suddenly and remain constant till ped=1. Therefore from the plot it is found that ped=0.80 would be optimum value for optimization.


MSE
0.675
0.67
0.665
0.66
0.655
0.65
0
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
Probability of Elimination Dispersal(ped)


MSE
0.68
0.67
0.66
0.65
0.64
1
2
3
4
5
Elimination Dispersal Step(Ned)


MSE
0.69
0.68
0.67
0.66
0.65
0.64
1
2
3
4
5
6
Reproduction Step(Nre)


MSE
0.68
0.678
0.676
0.674
0.672
0.67
0.668
0.666
0.664
0.662
4
8
12
16
20
Chemotactic Step(Nc)

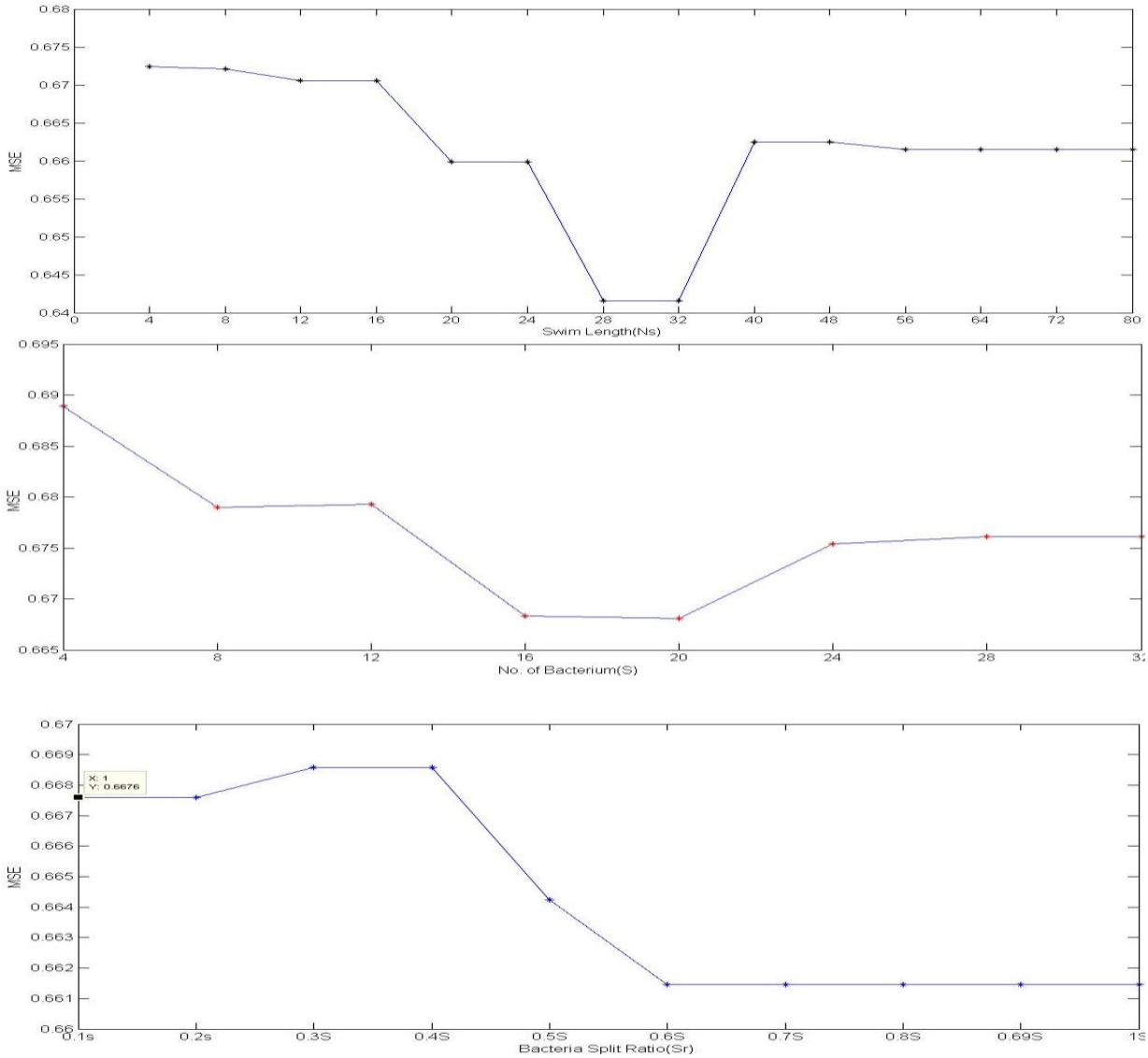


Figure 3.18: Plot between MSE v/s initial values parameters (a) ped (b) Ned (c) Nre (d) Nc (e) Ns (f) S (g) Sr

CHAPTER 5

RESULTS AND DISCUSSION

5.1 Similarity Measure

A colour video consisting of T frames and $M \times N \times 3$ array of pixel at locations (x, y) may be viewed as a "stack" of three scale frames corresponding to RGB components. The mean square error (MSE) is selected as the measure of performance as defined above.

$$MSE(f,I) = \frac{\sum_{t=1}^{T}\sum_{z=1}^{3}\sum_{x=1}^{N}\sum_{y=1}^{M}\left[I(x,y,z,t) - f(x,y,z,t)\right]^2}{3 \times N \times M}$$

Where I is the original colour video, f is the noisy video or the filtered video of size $N \times M$. Mean Square Error (MSE) gives a similarity measure between two videos. Another similarity measure is PSNR (peak signal to noise ratio) which is related to MSE as in the following:

$$PSNR(f,I) = 10\log\left(\frac{1}{MSE(f,I)}\right) \quad (40)$$

In Equation (38), MSE values are used in the normalized form. For example, if the frame class is uint8 then it has to be divided by 255*255. To be able to judge the performance of the proposed method, we will use the peak-signal-to- noise ratio (PSNR) as objective measures of similarity and dissimilarity between a filtered frame f and the original one I, each containing N rows and M columns of pixels. Higher the value of PSNR better is the similarity between two videos.

5.2 Comparison to Other State-of-the-Art Filters

In this section the performance of the proposed method is compared to that of Fuzzy Random Impulse Noise Removal from Color Image Sequence (FRINRFCIS) [65], video KNNF (K-nearest neighbour filter) [66], which was implemented using the 3D window; and video alpha trimmed mean (VATM) filter [66].

Fuzzy Random Impulse Noise Removal from Color Image Sequence [65] presented a filtering framework for color videos corrupted by random valued impulse noise in which the noise is removed step by step. The detection of noisy color components is based on fuzzy rules in which information from spatial and temporal neighbors as well as from the other color bands is used. Detected noisy components are filtered based on block matching where a noise adaptive mean absolute difference is used and where the search region contains pixels blocks from both the previous and current frame.

The video alpha trimmed mean (VATM) filter [66] performs both spacial and temporal filtering of image sequence by sorting pixels within a 3D window in ascending order and averaging a certain number of pixels in the window, depending of the optimal parameter value. The α-trimmed mean filter has to compromise between preservation of spatio-temporal details and efficient noise removal. In case of average noise levels ($p \approx 10\%$), at least half of the total number ($N=27$) of pixel values have to be averaged for efficient noise removal. However this will also introduces spatio-temporal blur into the sequence. As the noise level increases, efficient noise removal will be done at the expense of a higher spatio-temporal blur, further degrading the sequence structures. In case of medium or fast motion in the processed video sequence, more than a half of the pixel values, within the 3D sliding window, can belong to an object different from the one in the current frame. Consequently, considerable motion blur can be introduced.

Video K-nearest neighbour filter (KNNF) [66] is a non linear filter which sorts pixels within a 3D window, according to their difference with the central pixel's value; after that, it averages the pixels in the window, after weighting them according to their sorting order. Since for the specific noise type and noise level, the number of pixels considered for the averaging is constant, the video K-nearest neighbour filter can introduce certain amount of spatial and/or temporal blurring in case of fine spatial details and/or motion. This happens when there are not enough pixel values similar to the central pixel one, within the 3D sliding window area. Specifically, for efficient noise removal, at least half of the pixel values have to be used for filtering. As a result, in case of medium and fast motion some pixel values (from frames different from the current one), belonging to the undergoing motion object will also be in the averaging. Consequently, motion blur will be introduced.

We have tested the proposed methods on the following color image sequences : “Salesman”, “Tennis”, “Bus”, “Foreman”, “Flower”, and “Deadline” for random impulse noise level ranging from p = 5% to p = 10%. The result of this experiment in terms of PSNR is presented in fig 5.1. From the figure it can be concluded that the proposed method outperforms all other methods.

The proposed method outperforms these methods not just in terms of objective measure but also in visual comparison. Figs. 5.2, 5.3 and 5.4 respectively show for the 30th frame of the “Tennis” (p = 5%) sequence, 30th frame of the “Bus” (p = 10%) sequence and the 30th frame of the “Salesman” (p = 20%) sequence, the original frame, the noisy frame and the result obtained by the different compared methods.

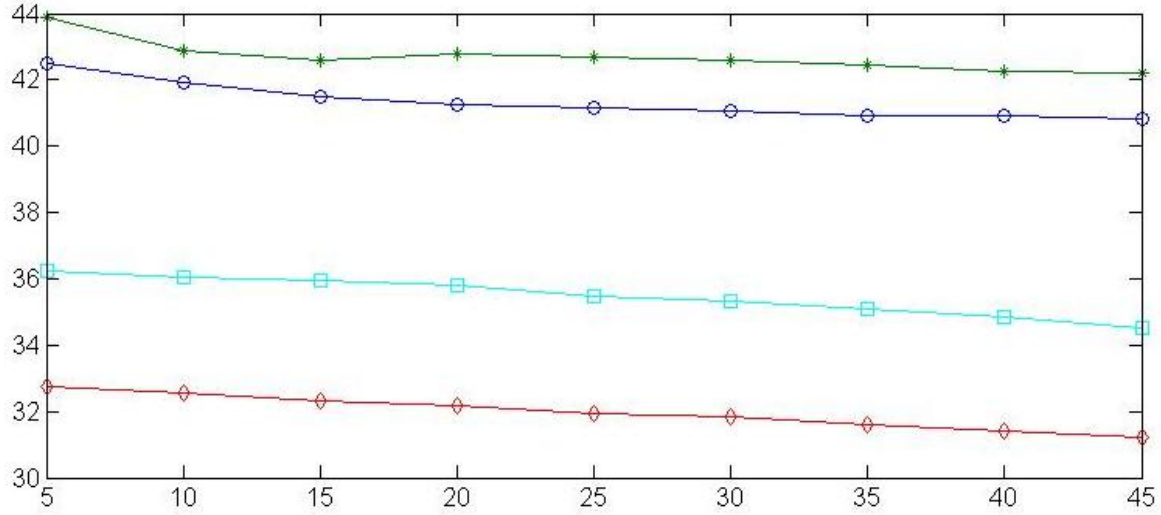
We see that the FRINRFCIS results in very good noise removal, even for high noise levels. At the cost of this, however, too much detail gets lost (e.g., railing in “Bus”) and the images become a little blurry. Further, several temporal inconsistencies in non-moving areas can be detected, especially when they are detailed (e.g., background leaves in “Salesman”).

VATM removes the noise very well. However, too many noise-free pixels are filtered as well, which results in both spatial and temporal inconsistencies, especially around edges. Further, the filter also performs less well in the case of motion (e.g., “Salesman” (face), “Tennis” (ball), “Bus” (car)). Since noisefree pixels are modified along with the noisy pixels, this filter results in blurring of the overall sequence.

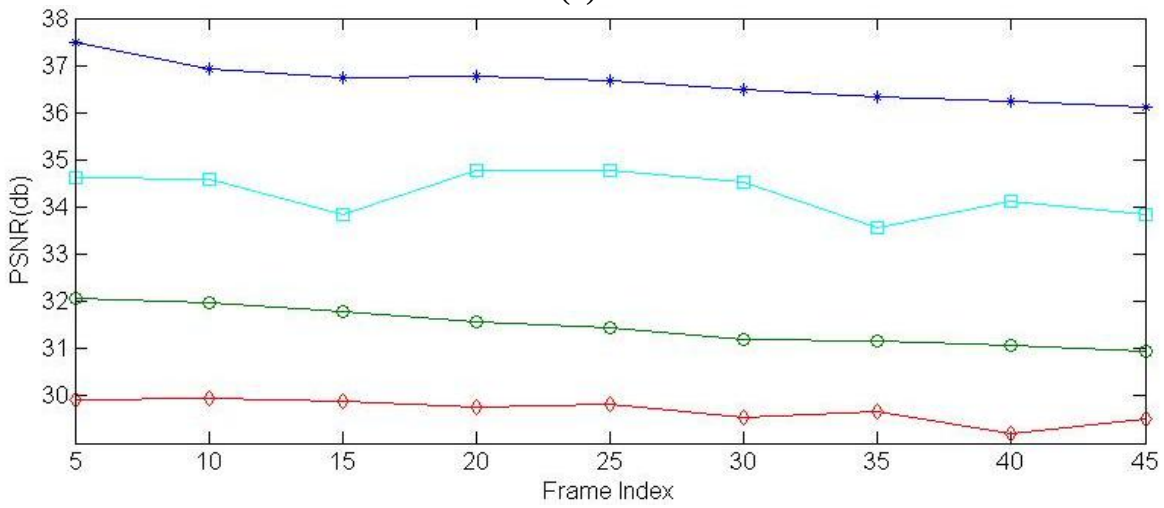
Similarly the KNNF removes the noise very well but introduces certain amount of spatial and/or temporal blurring in case of fine spatial details and/or motion (e.g., “Salesman” (arms), “Tennis” (ball), “Bus” (car)). Since a large number of pixels are used for computing the average, the filter produces blur even for low noise presence.

Finally, the proposed fuzzy filter combines very good detail preservation to very good noise removal and clearly outperforms all compared filters. The filter benefits very well from the extra information coming from similar regions in a spatio-temporal neighborhood.

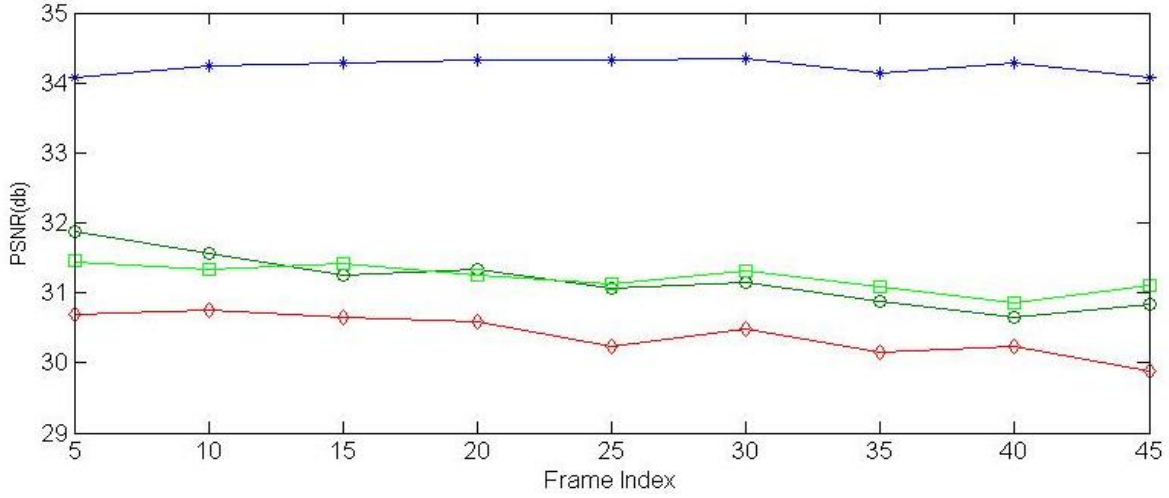
(a)


44
42
40
38
36
34
32
30
5
10
15
20
25
30
35
40
45

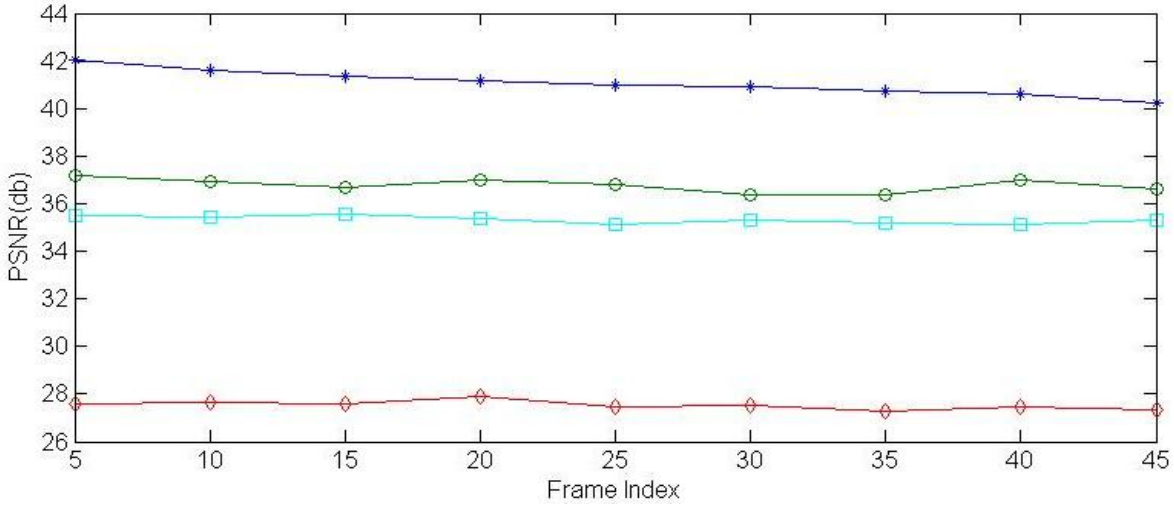
(b)


PSNR(db)
38
37
36
35
34
33
32
31
30
5
10
15
20
25
30
35
40
45
Frame Index

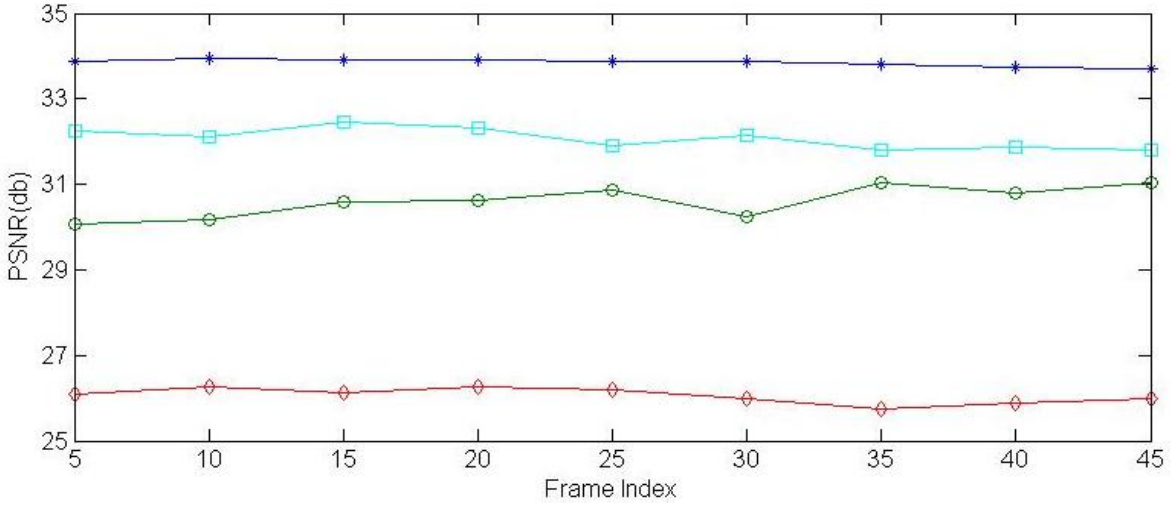
(c)


PSNR(db)
35
34
33
32
31
30
29
5
10
15
20
25
30
35
40
45
Frame Index

(d)


PSNR(db)
Frame Index
44
42
40
38
36
34
32
30
28
26
5
10
15
20
25
30
35
40
45

(e)


PSNR(db)
Frame Index
35
33
31
29
27
25
5
10
15
20
25
30
35
40
45

(f)

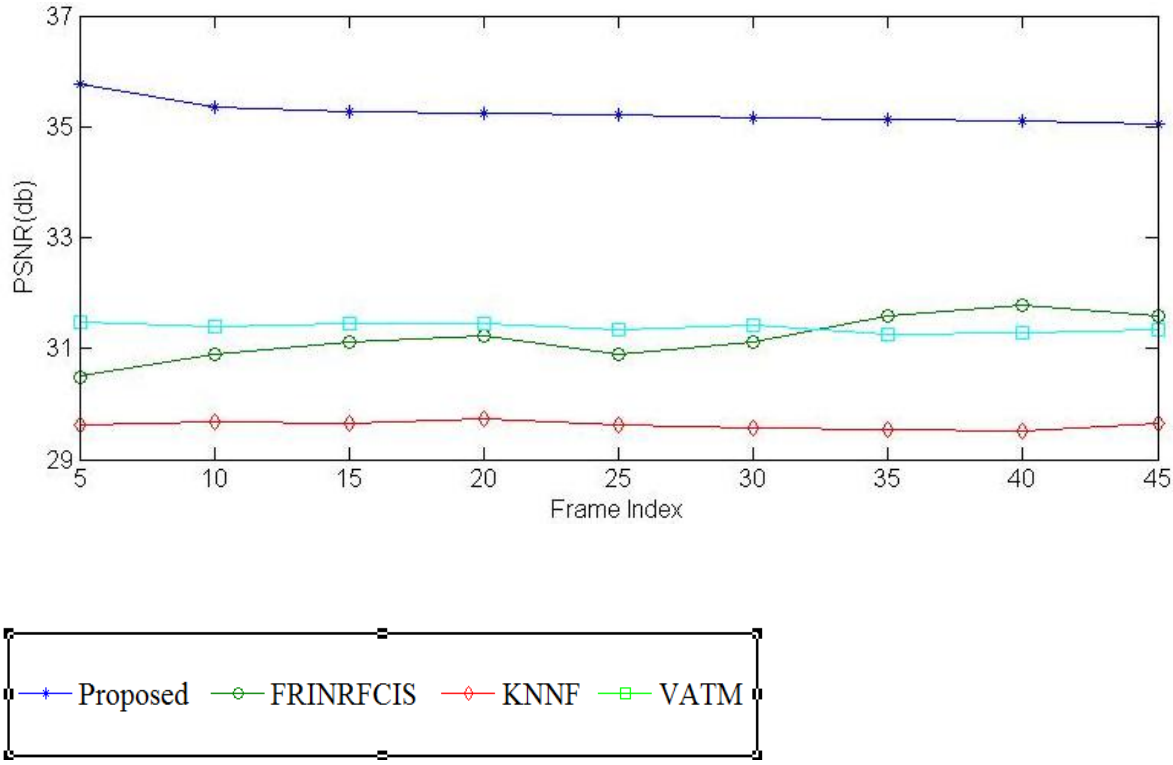


Figure 5.1 PSNR results for the different methods applied on (a) “Salesman” (p = 5%), (b) “Tennis” (p = 10%), (c) “Bus” (p = 15%), (d) “Foreman” (p = 20%), (e) “Flower” (p = 25%), (f) “Deadline” (p = 30%).

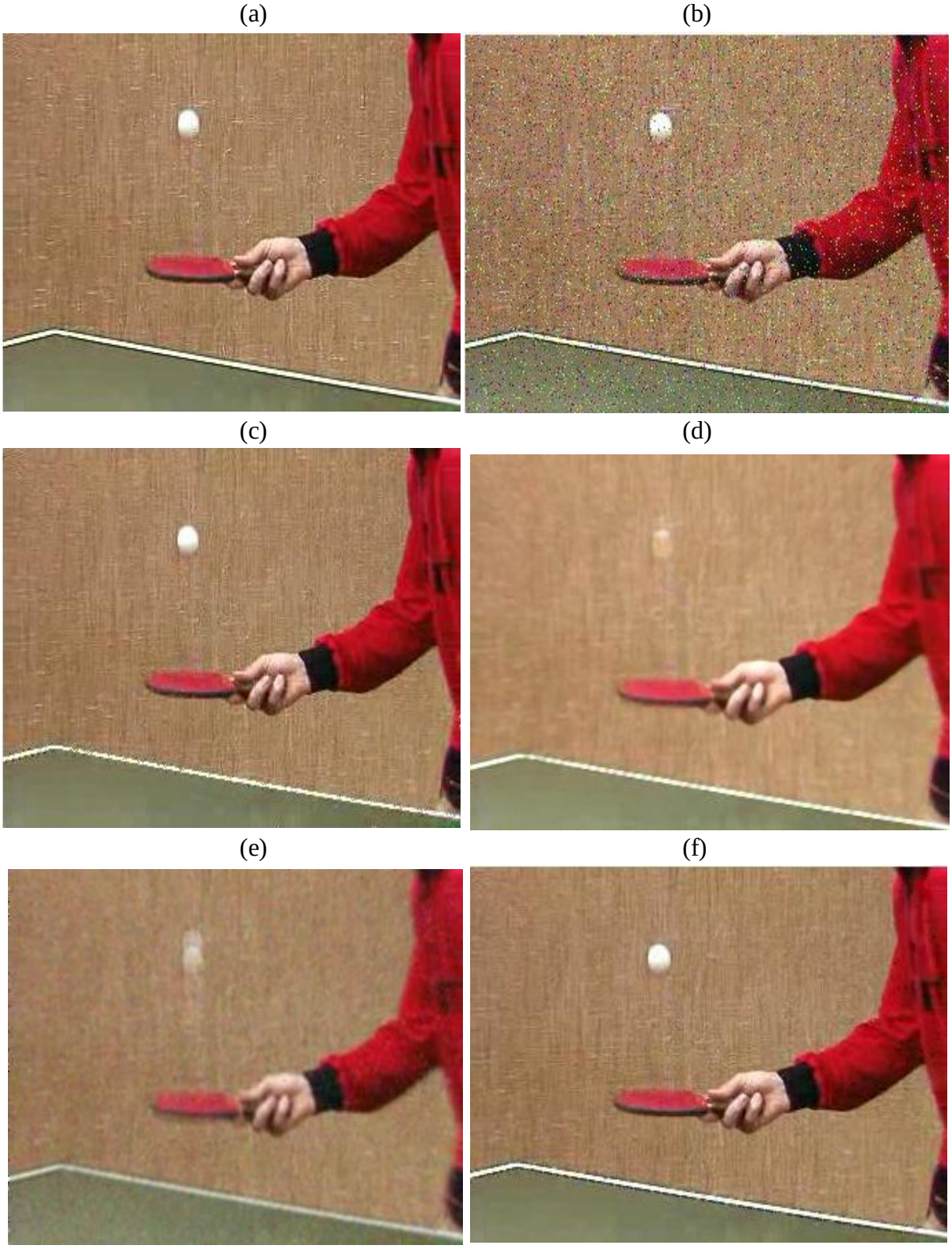


Figure 5.2: 30th frame of the "Tennis" sequence: (a) original, (b) noisy (p=5%), (c) FRINRFCIS, (d) VATM, (e) KNNF and (f) Proposed.

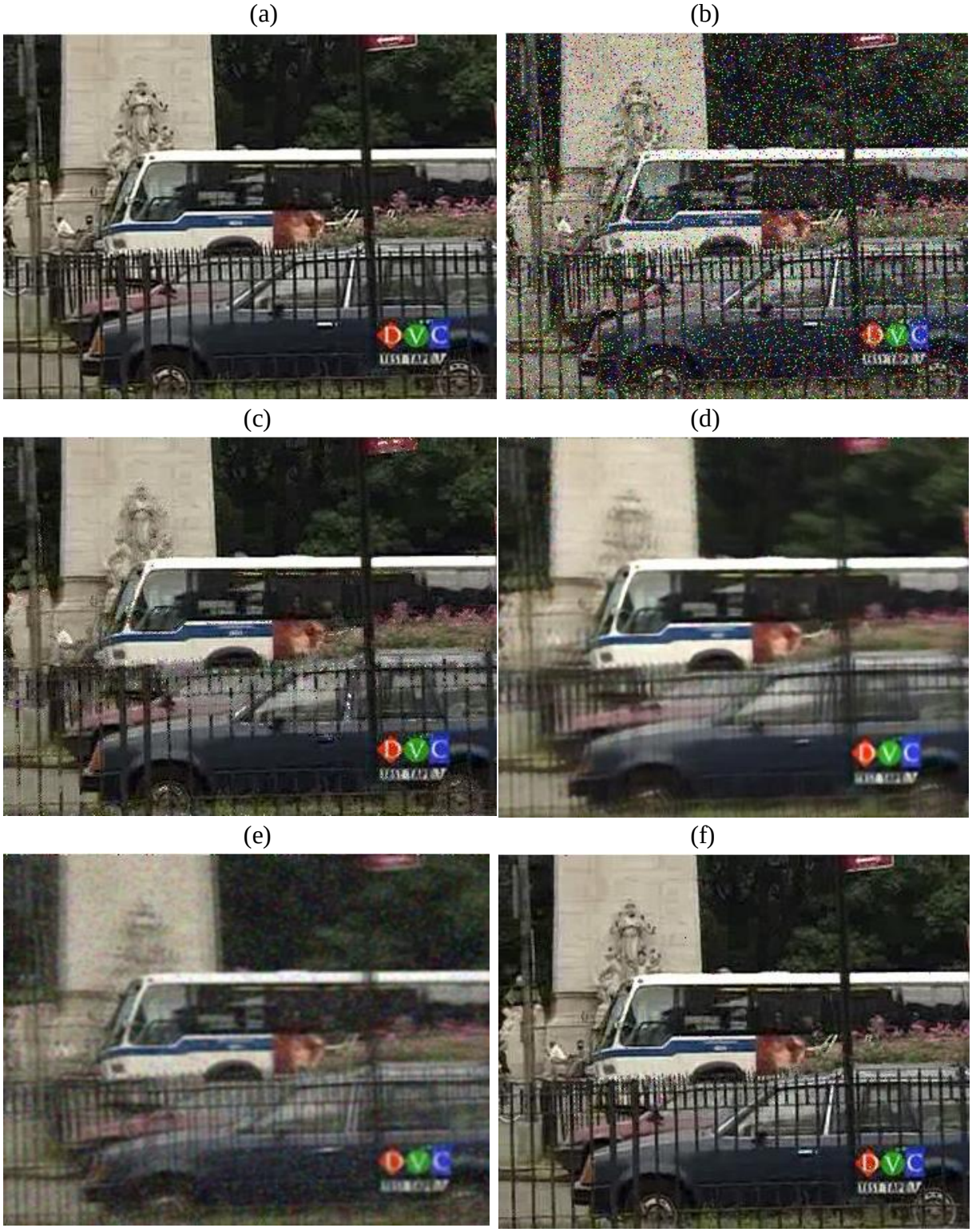


Figure 5.3: 30th frame of the "Bus" sequence: (a) original, (b) noisy (p=10%), (c) FRINRFCIS, (d) VATM, (e) KNNF and (f) Proposed.

Figure 5.4: 30th frame of the "Salesman" sequence: (a) original, (b) noisy (p=20%), (c) FRINRFCIS, (d) VATM, (e) KNNF and (f) Proposed.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

In this thesis, we have presented a new filtering framework for color videos corrupted by random valued impulse noise. In order to preserve the details as much as possible, the noise is removed step by step. The detection of noisy color components is based on fuzzy rules in which information from spatial and temporal neighbors as well as from the other color bands is used. Each of the color bands is filtered separately. However, the fuzzy rules that are used to determine the noisy and noisefree pixels in each step do now not only require information from a spatio-temporal neighborhood in the same color band, but exploit also the extra information that is available from the other color bands. To exploit the temporal information as well as the spatial information, we developed the technique that not only considers the current frame but also the previous and next frame. Two fuzzy sub filters are successively used to detect and filter the random impulse noise. Since a considerable part of the noise has already been filtered in the first step, the remaining noise is easier to detect in the second step and hence details are well preserved. Also, more reliable neighbors are used for comparison. The detection phase helps in determining the noisy pixels and in distinguishing them from the noisefree pixels. In this way, only noisy pixels are modified and noisefree pixels are left untouched which minimizes the blurring. Furthermore, only noisefree pixels are used for filtering the noisy pixels. In order to achieve good detection characteristics of the proposed method, it was necessary to optimize two types of parameters: the parameters in the original objective function and the parameters of the Bacteria Foraging entering into the objective function for facilitating the optimization. Both of the parameters are optimized in the present work. After the optimization, the proposed method is characterized by excellent results and robustness in the environments with a wide range of the impulse noise corruption. From the experimental results it can be seen that the proposed filters result in a very good trade-off between noise removal and detail preservation. They are further also shown to outperform all other compared state-of-the-art random impulse noise filters. The experiments show that the proposed method outperforms other state-of-the-art methods both visually and in terms of objective quality measures such as the PSNR.

6.2 Future Work

Image sequences play an important role in today's world. They provide us a lot of information. Videos are for example used for traffic observations, surveillance systems, and autonomous navigation and so on. Due to bad acquisition, transmission or recording, the sequences are however usually corrupted by noise, which hampers the functioning of many image processing techniques. The most common noise types that can be distinguished are impulse noise, additive noise and multiplicative noise. In the case of impulse noise, a certain percentage of the pixel grey values or color components are replaced by noise values. Such noise value can be fixed (usually as the minimum or maximum allowed value: salt and pepper noise) or the result of a random process (usually with a uniform distribution). If an image is corrupted by additive noise, then a random value from a given distribution (e.g. a Gaussian distribution) has been added to each pixel. In the multiplicative noise type, the intensity of the noise value added to a pixel depends on the intensity of the pixel grey value or color component itself (e.g. speckle noise). The proposed fuzzy filter is intended for color videos corrupted by random valued impulse noise and results in a very good trade-off between noise removal and detail preservation. As a future work we will try to extend this work for the denoising of color video sequences corrupted with other types of noise found in video application such as such as Gaussian noise and speckle noise.

Although the present work preserves the detail very well but does not include motion compensation as a pre-processing step. Motion compensation is a technique used in video compression, which is already adopted in video filters for additive Gaussian noise, but has not really found its way to impulse noise video filters yet. Hence, a possible direction for future work could be an extension of the proposed method to integrate simultaneous motion estimation technique into the present work.

REFERENCES

[1] Pawan Patidar, Manoj Gupta, Sumit Srivastava, and Ashok Kumar Nagawat, “Image De-noising by Various Filters for Different Noise”, International Journal of Computer Applications (09758887) Volume 9– No.4, November 2010.

[2] Mr.Salem Saleh Al-amri,Dr.Khamitkar S.D, “A comparative study of Removal Noise from Remote sensing Image”, IJCSI International Journal of Computer Science Issues,Vol.7,Issue .1,January 2010 32 ISSN (online):1694-0784 ISSN(Print):1694-0814.

[3] Om Prakash Verma, Madasu Hanmandlu, Anil Singh Parihar and Vamsi Krishna Madasu, “Fuzzy Filters for Noise Reduction in Color Images”, ICGST-GVIP Journal, Volume 9, Issue 5, September 2009.

[4] Fabrizio Russo, Giovanni Ramponi, “Nonlinear fuzzy operators for image processing”, Signal Processing, v.38 n.3, p.429-440, Aug. 1994.

[5] A. Taguchi, H. Takashima and Y. Murata, "Fuzzy filters for image smoothing", in Proc. SPIE Conf. on Nonlinear Image Processing V, San Jose, CA, USA, pp.332-339, February 7-9, 1994,.

[6] A. Taguchi, H. Takashima, F. Russo, "Data Dependent Filtering Using the Fuzzy Inference", in Proc. IEEE IMTC/95, Waltham, Mass., USA, April 24-26, 1995.

[7] S. Peng and L. Lucke, "Fuzzy filtering for mixed noise removal during image processing", in Proc. FUZZ-IEEE '94, Orlando, FL, USA, June 26 - 29,1994.

[8] S. Peng and L. Lucke, "Multi-Level Adaptive Fuzzy Filter for Mixed Noise Removal", in Proc. ISCAS-95, Seattle, Washington, USA, April 1995.

[9] S. Peng and L. Lucke, "An Adaptive Window Hybrid Filter", Proc. IEEE NSIP-95 Workshop, Neos Marmaras, Halkidiki, Greece, June 20-22, 1995.

[10] S. Peng and L. Lucke, "A Hybrid Filter for Image Enhancement", in Proc. IEEE ICIP-95, October 22-25, 1995, Washington, DC, USA.

[11] K. N. Plataniotis, D. Androutsos and A. N. Venetsanopoulos: "Color image processing using fuzzy vector directional filters", in Proc. IEEE NSIP-95, Neos Marmaras, Halkidiki, Greece, June 20-22.

[12] V. Chatzis and I. Pitas, "Mean and median of fuzzy numbers", Proc. IEEE NSIP-95, Neos Marmaras, Halkidiki, Greece, June 20-22, 1995.

[13] Pao-Ta Yu and Rong Chung Chen, "Fuzzy Stack Filters - Their definitions, fundamental properties and application in image processing", in Proc. IEEE NSIP-95, Neos Marmaras, Halkidiki, Greece, June 20 22.

[14] Tamalika Chaira, Ajoy kumar Raj "Fuzzy image Processing and Applications withMATLAB" CRC Press 2010.

[15] T A. Nodes and N. C. Gallagher, Jr. “The output distribution of median type filters,” IEEE Transactions on Communications, COM-32, 1984.

[16] T. Chen and H. R. Wu, “Space variant median filters for the restoration of impulse noise corrupted images,” IEEE Transactions on Circuits and Systems II, 48 (2001), pp. 784–789.

[17] H. Hwang and R. A. Haddad, “Adaptive median filters: new algorithms and results,” IEEE Transactions on Image Processing, 4 (1995), pp. 499–502.

[18] J. C. Brailean, R. P. Kleihorst, S. N. Efstratiadis, A. K. Katsaggelos, and R. L. Lagendijk. “Noise reduction filters for dynamic image sequences : A review”. Proceedings of the IEEE, 83(9):1236–1251, September 1995.

[19] J.S. Lee, ”Digital image enhancement and noise filtering by use of local statistics”, IEEE Trans. Patt. Anal. Machine Intell., vol. 2, pp. 165-168, 1980.

[20] J.S. Lee, ”Digital image smoothing and the sigma filter”, Computer Vision, Graphics and Image Processing, vol. 24, pp. 255-269, 1983.

[21] L.P. Yaroslavsky, ”Digital Picture Processing - An Introduction”, Springer Verlag, 1985.

[22] J. Tukey, ”Exploratory Data Analysis”, Addison-Wesley, 1977.

[23] A.K. Jain, "Foundimentals of Digital Image Processing", Englewood Cliffs, NJ, Prentice-Hall, 1989.

[24] C.Q. Zhan and LJ. Karam, "Wavelet-Based Adaptive Image Denoising With Edge Preservation", Proc. of the IEEE 2003 Int. Conf. on Image Process., MA- L2, Sept. 2003.

[25] Buades, B. Coll, J.M. Morel, "Denoising image sequences does not require motion estimation," avss, pp.70-74, IEEE Conference on Advanced Video and Signal Based Surveillance, 2005.

[26] M.B. Alp, and Y. Neuvo, "3-dimensional median filters for image sequence processing", IEEE Proc. Int. Conf. Acoustics, Speech, and Signal Proc., vol. 4, pp. 2917-2929, 1991.

[27] G. R. Arce, "Multistage order statistic filters for image sequence processing" IEEE Trans. Signal Processing, vol 39, pp. 1147-1163, 1991.

[28] A. C. Kokaram, "Motion Picture Restoration", PhD thesis, Cambridge University, 1993.

[29] B. Horn and B. Schunck, "Determining optical flow," Artif. Intell., Vol. 17, pp. 185-203, 1981.

[30] Davis, L., Rosenfeld, A.: "Noise cleaning by iterated cleaning". IEEE Trans. on Syst. Man Cybernet 8, 705-710 (1978).

[31] Mitchell, H., Mashkit, N.: "Noise smoothing by a fast k-nearest neighbor algorithm". Signal Processing: Image Communication 4, 227-232 (1992).

[32] Zlokolica, V.: "Advanced nonlinear methods for video denoising", PhD thesis, chapter 3, Ghent University, Ghent, Belgium (2006).

[33] Zlokolica, V., Philips, W.: "Motion-detail adaptive k-nn filter video denoising", Report (2002), http://telin.ugent.be/~vzlokoli/Report2002vz.pdf.

[34] Zlokolica, V., Pizurica, A., Philips, W.: "Video denoising using multiple class averaging with multiresolution". In: García, N., Salgado, L., Martínez, J.M. (eds.) VLBV 2003. LNCS, vol. 2849, pp. 172-179. Springer, Heidelberg (2003).

[35] Zlokolica, V.: “Advanced nonlinear methods for video denoising”, PhD thesis, chapter 5, Ghent University, Ghent, Belgium (2006).

[36] Cocchia, F., Carrato, S., Ramponi, G.: “Design and real-time implementation of a 3-D rational filter for edge preserving smoothing”. IEEE Trans. on Consumer Electronics 43(4), 1291-1300 (1997).

[37] Yin, H.B., Fang, X.Z., Wei, Z. and Yang, X.K., “An improved motion-compensated 3-D LLMMSE filter with spatio-temporal adaptive filtering support”. IEEE Trans. Circuits Syst. Video Technol. v17 i12. 1714-1727.

[38] Guo, L., Au, O.C., Ma, M. and Liang, Z., “Temporal video denoising based on multihypothesis motion compensation”. IEEE Trans. Circuits Syst. Video Technol. v17 i10. 1423-1429.

[39] L. A. Zadeh, "Fuzzy sets," Information and Control, vol. 8, no. 3, pp. 338-353, June 1965.

[40] Alper Pasha "Morphological image processing with fuzzy logic" Havaclilik ve uzayteknolojilerl derglsl ocak 2006 cilt 2 sayi 3 (27-34).

[41] Mamdani, E.H. and S. Assilian, "An experiment in linguistic synthesis with a fuzzylogic controller," International Journal of Man-Machine Studies, Vol. 7, No. 1, pp. 1-13,1975.

[42] Zadeh, L.A., "Outline of a new approach to the analysis of complex systems anddecision processes," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 3, No.1,pp. 28-44, Jan. 1973.

[43] Arshdeep Kaur, Amrit Kaur, ”Comparison of Mamdani-Type and Sugeno-Type Fuzzy Inference Systems for Air Conditioning System,” International Journal of Soft Computing and Engineering (IJSCE) , ISSN: 2231-2307, Volume-2, Issue-2, pp. 323 – 325, May 2012 ISSN: 2231-2307, Volume-2, Issue-2, May 2012.

[44] L.N.Decastro, F. J. Von Zuben, and Idea Group Pub et al. “Recent Developments In Biologically Inspired Computing”. IGI Publishing, Hershey, PA, USA, 2004.

[45] D.E. Goldberg, “Genetic algorithms in search, optimization and machine learning”, Addison-Wesley, Reading, MA, 1989.

[46] Poli R.: “Genetic Programming for Image Analysis”. Proceedings of the First International Conference on Genetic Programming, 363-368, 1996.

[47] Konstantinos Delibasis , Peter E. Undrill , George G. Cameron, “Designing texture filters with genetic algorithms: an application to medical images”, Signal Processing, v.57 n.1, p.19-33, Feb. 1997.

[48] Szostakowski J. “Genetic programming for Image Sequence Filtering”. Proc. International Conference on Power Electronics and Intelligent Control for Energy Conservation, Warsaw, 2005.

[49] Lukac R., Plataniotis K. N., Smolka B., Venetsanopoulos A.N.: “Color image filtering and enhancement based genetic algorithms”. Proc. International Symposium Circuits and Systems, 913-916, 2009.

[50] Lee C. S., Guo S. M., Hsu C. Y. (2005) “Genetic-based fuzzy image filter and its application to image processing”. IEEE Transactions on Systems, Man, and Cybernetics, Part B, Cybernetics 35(4): 697–711.

[51] Eberhart, R.C., Kennedy, J.: “A new optimizer using particle swarm theory”. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science. Kluwer Academic Publishers, Nagoya, Japan (1995) 39-43.

[52] Shi, Y. and R. Eberhart, 1998. “A modified particle swarm optimizer. Proceeding of the Congress on Evolutionary Computation”, May 4-9, IEEE Xplore Press, Anchorage, AK, USA, pp: 69-73.

[53] Kennedy, J. and Eberhart, R. C. “Particle swarm optimization” Proceedings of IEEE International Conference on Neural Networks Vol. IV, IEEE service center, Piscataway, NJ, Nov. 27- Dec. 1, 1995, pp. 1942-1948.

[54] M. Dorigo and G. Di Caro, "The Ant Colony Optimization meta-heuristic," in New Ideas in Optimization, D. Corne et al., Eds., McGraw Hill, London, UK, pp. 11–32, 1999.

[55] M. Dorigo, G. Di Caro, and L.M. Gambardella, "Ant algorithms for discrete optimization," Artificial Life, vol. 5, no. 2, pp. 137–172, 1999.

[56] M. Dorigo and T. Stutzle, "Ant Colony Optimization", MIT Press, Cambridge, MA, 2004.

[57] N. Meuleau and M. Dorigo, "Ant colony optimization and stochastic gradient descent," Artificial Life, vol. 8, no. 2, pp. 103–121, 2002.

[58] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents", Part B: Cybernetics, IEEE Transactions on Systems, Man, Cybernetics, 1996, Vol. 26, Issue 1, pp. 29–41.

[59] Passino, K.M., 2002. "Biomimicry of bacterial foraging for distributed optimization and control". Control Systems Magazine, IEEE 22 (3), 52–67.

[60] Liu, Y., Passino, K.M., 2002. "Biomimicry of social foraging bacteria for distributed optimization models, principles and emergent behaviors". J. Optim. Theory Appl. 115 (3), 603–628.

[61] Hanmandlu M., Verma O. P., Kumar N. K., Kulkarni M. (2009) "A novel optimal fuzzy system for color image enhancement using bacterial foraging". IEEE Transactions on Instrumentation and Measurement 58(8): 2867–2879.

[62] Kim D. H., Abraham A., Cho J. H. (2007) "A hybrid genetic algorithm and bacterial foraging approach for global optimization". Information Sciences 177(18): 3918–3937.

[63] Biswas A., Dasgupta S., Das S., Abrahamy A. (2007) "A synergy of differential evolution and bacterial foraging algorithm for global optimization". International Journal on Neural and Mass-Parallel Computing and Information Systems, Neural Network World 17(6): 607–626.

[64] Mishra, S., "A hybrid least square-fuzzy bacteria foraging strategy for harmonic estimation," IEEE Trans. Evol. Comput., Vol. 9, No. 1, 61–73, Feb. 2005.

[65] Tom Mélange, Mike Nachtegael, Etienne E. Kerre: “Fuzzy Random Impulse Noise Removal From Color Image Sequences”, IEEE Transactions on Image Processing 20(4): 959-970 (2011).

[66] Zlokolica, V., Philips, D., and van de Ville, D.: “A new non-linear filter for video processing". IEEE Benelux Signal Processing Symp. (SPS-2002), Belgium, March 2002.