Video Transmission Jerkiness Measure

Deena Abdelsamad

This thesis is presented as part of Degree of
Bachelor of Science in Electrical Engineering

Blekinge Institute of Technology
June 2012

Blekinge Institute of Technology
School of Computing
Supervisor: Raja M. Khurram Shahzad
Examiner: Benny Lovstrom

Abstract

Digital video transmission is widely used nowadays in multimedia. Frame dropping, freeze and reduced number of frames in the transmitted video are common symptoms of bad transmission quality. In order to assess the quality of transmission, a criterion is introduced in a model for a no reference video jerkiness measure [3]. This model is different from the former models presented as it depends on viewing conditions and video resolutions, so it is applicable for any frame size from QCIF to HD. The model uses simple mathematical equations of jerkiness and can be used for any video sequence [3]. A model of reduced reference method (Qtransmission) which depends on a pre-measured Jerkiness is introduced as a suggestion of future work.

The algorithm of video jerkiness measure model [3] is used and a MATLAB program is implemented to measure the jerkiness of the transmitted video. The program is used to calculate the jerkiness of QCIF video resolution in both (avi) and (yuv) formats. Twelve test cases are used to test the implemented program; the first six test cases are in (avi) format while the remainder cases are in (yuv) format. The first nine cases are error free with no observed frame freeze or drop. Intentional error (freeze) is added in the last three test cases, frame freeze is forced to occur in almost a third of the video frames and the jerkiness is calculated accordingly. The implemented program is used to calculate the video jerkiness, conclusion and results are presented and discussed in chapter 4 of this thesis.

Acknowledgments

Completion of work gives a nice feeling of happiness and success. The accomplishment of this thesis gives me the confidence to move ahead and teaches me a new way of knowledge and learning. I would like to convey my gratefulness, praise and thanks to ALLAH (swt) the most gracious and the most merciful, who blessed me with his grace and helped me to write this thesis successfully. Thanks to Anders Hultgren, program manager and Raja M. Khurram Shahzad my supervisor who provided me with valuable comments and suggestions for improvement. Thanks are due to all the BTH staff members who are always friendly and supportive. Last but not least, my sincere gratitude goes to my parents and my husband for their continuous mental, emotional and financial support. I feel lucky and proud to be part of my family.

Contents

List of Figures

List of Tables

List of abbreviations

1. HD - High-definition
2. SD - Standard-definition
3. CIF - Common intermediate format
4. QCIF - Quarter common intermediate format
5. AVI - Audio Video Interleave
6. CRT - Cathode ray tube
7. VCR - Video Cassette Recording
8. LCD - Liquid-crystal display
9. DLP - Digital Light Processing
10. VQEG - Video Quality Experts Group
11. DVD - Digital Versatile/Video Disc
12. MPEG - Moving Picture Experts Group
13. VHS - Video Home System
14. PAL - Phase Alternating Line
15. NTSC - National Television System Committee
16. 3D - Three-dimensional
17. PDP - Plasma display panel
18. RGB - Red, Green, Blue
19. SECAM - Sequentiel couleur a memoire (French for Sequential Color with Memory)

Chapter 1

Introduction

Digital videos transmitted over communication networks are easily subjected to frame dropping, freezing and reducing. For an end user these impairments cause a non-smooth presentation of the video that is called Jerkiness. A general definition of frame freezing is given below.

"Frame freezing can be caused by low-bit rate encoding when there is an overflow in the encode buffer. It can also be caused by the transmission of the video stream over an error prone channel" [4].

Frame skipping affects image rate freezing and jerky motion especially at low bitrates. This also affects the video quality at the final receptor causing image sharpness loss such as blur and impairments such as freezing and jerkiness. If the final receptor is human, it is important to have a no-reference measure for the received video quality to detect Jerkiness from the output of a video player. This measure depends only on information taken from the output video sequence where no information about the video before encoding or transmission is required [9].

In a study, a model for measuring jerkiness is proposed [3]. Results indicated that the proposed model is unique and better from existing models of that time [3]. First, it is applicable for any frame size from quarter common intermediate format (QCIF) to high definition (HD), that depends on the viewing conditions and video resolution. Second, the model can be customized and used for any video sequence. Third, simple mathematical form is used to measure the Jerkiness. Model of calculating Jerkiness is used to calculate the variables value, and then the variables are defined in the implemented program and substituted in the following formula [3].

$$J\left(v\right)=\frac{1}{T}\sum_{\infty}\triangle t_i\cdot\tau_\alpha\left(\triangle t_i\right)\cdot\mu\left(m_{i+1}\left(v\right)\right) \tag{1.0.1}$$

1.1 Scope

The scope of this thesis is to measure jerkiness of an output transmitted video by implementing an algorithm of a no-reference measure for the received video quality based on a no-reference model which depends only on information taken from the output video sequence [3]. A test will be run first on the implemented program to check it's efficiency to determine frame number and calculate motion intensity, later on, some video samples are tested with and without freeze to determine the freeze effect on jerkiness value.

1.2 Outline

Chapter 2 focuses on the basic concepts and background for image and video history. Chapter 3 describes the design and implementation of the program, it presents a description of MATLAB commands used and tested. Finally in Chapter 4, results are presented and their explanation is given. Chapter 4 also concludes the work and provides directions for future work.

Chapter 2

Background and related work

2.1 Background

2.1.1 The History of the Motion Picture

“Wheel of life“ or “Zoopraxiscope” was the first machine invented by William Lincolin in the United States in 1867 that was able to show animated photos and movies. The motion picture usage started by inventing the motion pictures cameras by Lumiere 1895 [1]. It included the three following functions, i.e., mobile motion-picture camera, projector and movie processing unit. Although some others had similar inventions at the same point of time but Lumiere and his brother were considered the first to present these technologies of photographic, projected and moving pictures to audience of one person or more [1]. A research team, led by Charles Ginsburg, at the Ampex Corporation was responsible to invent a recording magnetic video tape that can capture the live pictures from the TV camera and converts them into electrical pulses and saves them. The team succeeded to develop magnetic video tape in 1951 [1]. In 1971 the Sony company sold the first video cassette recording (VCR) to consumers [1]. Later, a massive decrease of video tapes sales took place by invention of digital video disc (DVD) in 1997 and the Blu-ray in 2006 and video tapes became an old fashion [1].

> “Vidre” is a Latin verb with the meaning “See”, while the word Video typically means “I See”. But practically it refers to store the moving photos with different digital formats like DVD, moving picture experts group (MPEG), audio video interleave (AVI) or analog formats like video home system (VHS) and transmits them with different techniques like phase alternating line (PAL) or national television system committee (NTSC) [1]. The quality of a transmitted video depends on different parameters such as how the moving pictures were captured and how they were stored. A modern format of television video became the standard format which can offer higher

quality than the former ones which is digital television (DTV) [6].

2.1.2 Number of frames per second

The frame rate is the number of pictures per unit of time in the video; it differs according to the type of the camera used to capture pictures. The normal rate of old cameras can vary from six to eight frames per second, while it reaches up to 120 frames per second for modern cameras. It is discovered that the frame rate affects the transferring process of the cinematic motion picture to a video film, as the movie recorded at slow frame rate like 24 photo-grams per second can make the transmission more complicated. Besides, to get the illusion of a video movie, without giving the feeling to user that he/she is watching moving photos, a frame rate of at least 15 frames per is suggested [7].

2.1.3 Interlacing

Video has two scan formats, progressive or interlaced, such as sequential color with memory (SECAM), NTSC and PAL 576i50. Interlaced video is a method to double the received frame rate displayed with the signal used with analog television without consuming more bandwidth [12]. Interlacing is indicated in the video data as “i” where 576 is the resolution of the vertical line and 50 is the fields of half-frames per second. The main usage of interlacing is to get the best video quality if the bandwidth is limited. In each interlaced frame, horizontal scan lines are numbered respectively and divided into two fields, i.e., upper field and lower field. The upper field also called “odd field” contains lines with odd numbers, the lower field that is also called “even field” contains the lines with even numbers. This interlaced stream like DVD or analog can be converted by a method called “deinterlacing” to use it with the progressive devices like liquid crystal display (LCD) and plasma screens. However, this deinterlacing method is unable to give the same video quality processed by progressive scan. The progressive system has a different technique, it updates all scan lines with every refresh period, which enhances the resolution and decreases errors like moving or flashing of the constant pictures [12].

2.1.4 RGB color model

RGB model is an additive color model where R stands for red, G for green and B for blue, light is added in different ways to reproduce a massive range of the colors. The main purpose of the RGB model is to display, represent and sense image in the electronic systems, for example computer screens and televisions [10]. RGB has different input methods such as video, television cameras and image scanners. RGB has different output devices as well such as mobile phones, computer screens, projectors and televisions with different systems like cathode ray tube (CRT), LCD and plasma [6].

2.1.5 YUV

"YUV" is a color space which encodes videos considering human perception. It also refers to the complete range of colors that can be recorded and displayed in a digital video [2]. Y is for Luma (brightness) component while U and V are for chrominance (color) components respectively. YUV color encoding system is used for PAL and NTSC. The equations to convert RGB to YUV are [2]:

$Y = 0.299R + 0.587G + 0.114B$
$U = 0.147R - 0.289G + 0.436B$
$V = 0.615R - 0.515G + 0.100B$

2.2 Related work

In the last few years, researches have been done on a non-reference video quality measure. Some of them presented a new metric to evaluate and detect the effect of image dropping on user quality perception [8]. That measure was based on a psycho-visual quality function and temporal summation function (temporal pooling) modeling the assessment mechanism of the human assessors [8]. This assessment model integrates the abrupt temporal variation that appears at the end of fluidity impairments as a second factor for quality estimation [8]. While other similar studies were done about the same type of measure, however, the measure is based on freezes, jerky motions and rate variations of an image [9]. The measure should show a significant correlation with the observers ratings in an attempt to reproduce some basic perceptual human visual process involved within the task of video quality assessment [9].

Chapter 3

Design and Implementation

3.1 Requirements

To measure the jerkiness of a video, a program is required that is capable of reading video frames and converts them to numerical data (Y, Cr, Cb). The program must also calculate the motion intensity and evaluate the required parameters to compute the jerkiness.

3.2 Methodology

In this thesis, the aim is to implement a no-reference measure of jerkiness that depends on information taken from the output video sequence where no information about the video before encoding or transmission is required [10].

Borer's model algorithm and formulas are used in calculating the variables values. Then, they are defined in the developed program and substituted in the following formula of calculating Jerkiness.

$$J\left(v\right) = \frac{1}{T}\sum_{\infty} \triangle t_i \cdot \tau_\alpha\left(\triangle t_i\right) \cdot \mu\left(m_{i+1}\left(v\right)\right) \tag{3.2.1}$$

Displayed images with certain rate and time stamps form a video sequence can be denoted by $v = \left(f_i, t_i\right), \quad i = 1...n$ [3]. Where f_i is frame "i" of the video, t_i is starting displaying time up to t_{i+1} and n is the total number of frames. In case that the number of frames is already known, it will be mentioned as time stamps $\triangle t_i = t_{i+1} - t_i$. It is assumed that calculating motion by the theory of calculating the statistic velocity distribution of all objects moves in the video sequence is very complicated, so a simpler measure is used for motion calculation in a video sequence $v = \left(f_i, t_i\right)$ by the following formula [3]. Where $f_i\left(x\right)$ is the pixel value of the Y-component of the frame "i" located at the x location.

$$m_{i+1}\left(v\right) = \sqrt{\sum_{\infty}\left(f_{i+1}\left(x\right) - f_i\left(x\right)\right)^2} \tag{3.2.2}$$

The jerkiness calculation in Borer's model [3] depends on the number of frames, frame display time $\triangle t_i$ and frame motion intensity. The dependency on $\triangle t_i$ and the motion intensity are expressed by two S-shaped (sigmoid functions) $\tau_\alpha$and μ. These functions have three parameters represented in position of x, position of y and slope of the inflection point. For calculating the motion dependent part μ, these three parameters are kept fixed. For the τ_α of display time dependent part τ they are re-parameterized by single α parameter where $a = p_y / p_x^{qp_x/p_y}$, $b = qp_x/p_y$, $c = 4q/d$ and $d = 2\,(1 - p_y)$.

$$s\,(x) = \begin{cases} ax^b & if\, x \leq p_x \\ \frac{d}{1+\exp(-c(x-px))} + 1 - d & else \end{cases} \tag{3.2.3}$$

As the algorithms are tested on QCIF resolution videos, so for μ the parameters are $(p_x, p_y, q) = (5, 0.5, 0.25)$ and for $\tau_\alpha$the parameters are $(p_x, p_y, q) = (0.12, 0.05, 1.5)$. The S-shaped function starts at the origin and increases polynomial until it reaches the inflection point then it saturates exponential towards one. The viewing angles and distances are proved to affect the results; tests are performed with the typical viewing angle and distance [5]. The viewing distance is equal to three times the height of the picture (3 H) and viewers are seated directly in line with the center of the video display. The resolution is set as QCIF and motion intensity is measured on the sub-sampled frames in case of larger resolutions.

3.3 Implementation

3.3.1 Reading code

The program for calculating video jerkiness consists of two parts. The main program is “ReadVid.m” that contains all the functions required to read the video, slice it into frames, compute motion intensity and other required values in order to compute the jerkiness. The second part is “Sshape.m”, it calculates the s-shape value required to computeμ, τ.

In figure 3.3.1, clear all command is used to clear the MATLAB screen, and then close all to close any open windows or figures opened in MATLAB. The code also performs the following:

Read video specified
slice video into frames
convert RGB matrices of each frame to YCbCr matrices

```
clear all
close all
infile = ’clip6.avi’;
readerobj = mmreader(infile)
vidFrames = read(readerobj);
numFrames = size(vidFrames,4);
rgb=zeros(1,1,1);
for k = 1 : numFrames
rgb = vidFrames(:,:,:,k);
% transfearing the rgb data to YCbCr
mov(k).cdata=rgb2ycbcr(rgb);
end
```

Figure 3.3.1: Read Video

```
info = mmfileinfo(infile);
hight=info.Video.Height;
width=info.Video.Width;
qcifflag=0;
cifflag=0;
hdflag=0;
sdflag=0;
c=0;
%QCIF 176 × 144
%CIF/SIF(625) 352 × 288
% HD width is >= 1080
if ((width==176)&&(hight==144))
qcifflag=1;
c=1;
end
if ((width==352)&&(hight==288))
cifflag=1;
c=1.18;
end
```

Figure 3.3.2: Resolution Parameters

In figure 3.3.2, “mmfileinfo” MATLAB function is used to get the frame width and height. Each resolution (QCIF, CIF, HD, SD) will have certain constants in the future calculations. Four flags (qcifflag, cifflag, hdflag, sdflag) are initialized to have the value of zero then according to its values of “width” and “height” the appropriate flag are set. For example if width =176 and height = 144 so the video is of QCIF resolution. Therefore, the qcifflag is set to 1 while other flags remain zero.

```
if ((hight==480)||(hight==576))
sdflag=1;
c=1.54;
end
if (width>=1080)
hdflag=1;
c=2.54;
end
if ((qcifflag==0)&&(cifflag==0)&&(sdflag==0)&&(hdflag==0))
disp(’the vedio resolution is not recognized’);
disp(’excution terminated’);
break;
end
```

Figure 3.3.3: Resolution Settings

Figure 3.3.3, continues to set the appropriate frame resolution flag. The last part of the code is to break the execution of the program if the frame resolution is not recognized.

```
% the mu parameters [Px,Py,q]
mupar=[5,0.5,0.25];
% the tau(alfa) =[Px/c, Py, q*c]
taupar=[0.12/c,0.05,1.5*c];
```

Figure 3.3.4: mu Parameters

In figure 3.3.4, as the code is written to measure the jerkiness of QCIF videos, so the parameters of μ, τ of QCIF videos are used. Note that c=1 for QCIF resolution.

```
% calculating the motion intensity
% for avi Dt is uniform
dt=info.Duration/numFrames;
for k=1:numFrames-1 %
extracting the R matrix of frames F(i) , F(i+1)
Fiy= double(mov(k).cdata(:,:,1));
Fip1y=double(mov(k+1).cdata(:,:,1));
Ficb= double(mov(k).cdata(:,:,2));
Fip1cb=double(mov(k+1).cdata(:,:,2));
Ficr= double(mov(k).cdata(:,:,3));
Fip1cr=double(mov(k+1).cdata(:,:,3));
irow=size(Fiy,1);
jcol=size(Fiy,2);
dumy=0;
dumcb=0;
dumcr=0;
for i=1:irow
for j=1:jcol
dumy=dumy+(Fip1y(i,j)-Fiy(i,j))^2;
dumcb=dumcb+(Fip1cb(i,j)-Ficb(i,j))^2;
dumcr=dumcr+(Fip1cr(i,j)-Ficr(i,j))^2;
end
end
dumy=double(dumy);
dumcb=double(dumcb);
dumcr=double(dumcr);
my(k)=sqrt(dumy)/width;
mcb(k)=sqrt(dumcb)/width;
mcr(k)=sqrt(dumcr)/width;
% mav=sqrt((dumr+dumg+dumb)/3);
end
figure(1)
plot(my,’r’);
hold on
%plot(mcb,’g’);
%hold on
%plot(mcr,’b’);
%hold on
%plot(mav,’k’);
title(’motion intesity’);
ylabel(’Motion intensity’);
xlabel(’Frame no’)
```

Figure 3.3.5: Motion Intensity Calculation

In figure 3.3.5, the code computes the motion intensity and plots it.

```
% computing the jerkness
imax=max(size(my));
dum=0.0;
for i=1:imax
mu(i)=Sshape(mupar,my(i));
tau(i)=Sshape(taupar,dt);
dum=dum+dt*tau(i)*mu(i);
end
jerkness=1.0/info.Duration*dum
dt
taupar
figure(2)
plot(mu,’r’);
title (’ mu values’)
ylabel(’mu’);
xlabel(’Frame no’)
figure (3)
plot(tau,’b’);
ylim([0.8*min(tau),1.2*max(tau)])
title (’Tau values’)
ylabel(’Tau’)
xlabel(’Frame no’)
disp
(’finish’)
```

Figure 3.3.6: Calculating Jerkiness

Figure 3.3.6, shows the part that calculates $\mu$and τ using Sshape function which is presented in the next section. The code uses the already derived $\mu$and τ to compute the jerkiness while MATLAB Figures (2,3) plot μ and τ respectively for each frame in the video.

3.3.2 Sshape Function

The Sshape function in figure 3.3.7, computes $\mu$and τ of a frame in the video sequence. It uses the $\mu$and τ parameters as an input plus $\triangle t$ if τ is required or motion intensity if μ is required. The returned value (μ or τ) is the “out” variable in figure 3.3.7.

```
function out = Sshape(invec,x)
px=invec(1);
py=invec (2);
q=invec(3);
b=q*px/py; d=2*(1-py);
c=4*q/d; a= (py/px)^b;
if x<=px out=a*x^b; else out=d/(1+exp(-c*(x-px) ))+1-d;
end
end
```

Figure 3.3.7: Sshape function

3.3.3 YUV to MOV Conversion

In figure 3.3.8, values are set and the freezing code is executed in the video, then the video is converted from YUV to MOV [11]. This part of the code should perform the following:

Delete any former memory in working space
Close all the opened windows
Read file clip9
start from 2/3
end at 3/3
Sample rate = 420
Convert YUV to MOV
Number of frames = size of matrix

```
clear all
close all
infile = ’clip9’;
clip=[2/3,3/3];
samplerate=420;
mov = yuv2mov([infile,’.yuv’],176,144,num2str(samplerate));
numFrames = size(mov,2);
hight=144;
width=176;
```

Figure 3.3.8: YUV to MOV Conversion

3.3.4 Adding Freeze

In figure 3.3.9, some freeze (repeated frames) is added to the video to compare the different results when the code runs on same video with and without freeze. This part of the code should perform the following:

Start freeze from istart
avoid decimals
Start from the first part or number 1

```
start from (2/3)
stop at (3/3)
Start the freezing from start to stop
K = number of the current frame used
delete the values of frames from start to stop
repeat frame of start
End
```

```
istart=round(1+clip(1)*numFrames);
iend=round(clip(2)*numFrames);
for k=istart:iend
mov(k).cdata=mov(istart).cdata;
end
```

Figure 3.3.9: Adding Freeze

3.3.5 MOV to YUV Conversion

In figure 3.3.10, this function is used to convert MOV object to YUV file [11]. The code takes the old name of the freeze free file and add to it the word “-freez”, then it takes the same sample rate in a string using the MATLAB tool number to string.

```
mov2yuv([infile,’-freez’,’.yuv’],mov,num2str(samplerate));
```

Figure 3.3.10: MOV to YUV Conversion

3.4 Testing

To test the code, freezing algorithm is used to freeze certain parts in the video. It is expected that the video with some freezing frames gives lower jerkiness value than the video without freezing. A simple test is made which freezes the entire video; it contains one frame but repeated 300 times so that all the frames have the same numerical data. The frame repetition leads to zero motion intensity for all frames and zero jerkiness as well. The implemented program gives the zero jerkiness value as expected.

Chapter 4

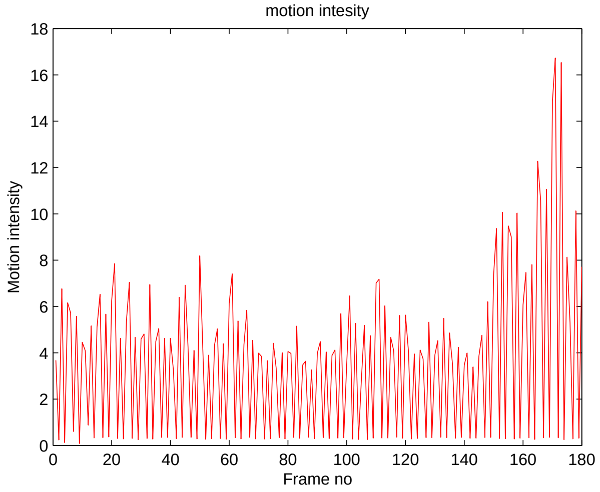
Results, Conclusions and Future work

4.1 Results

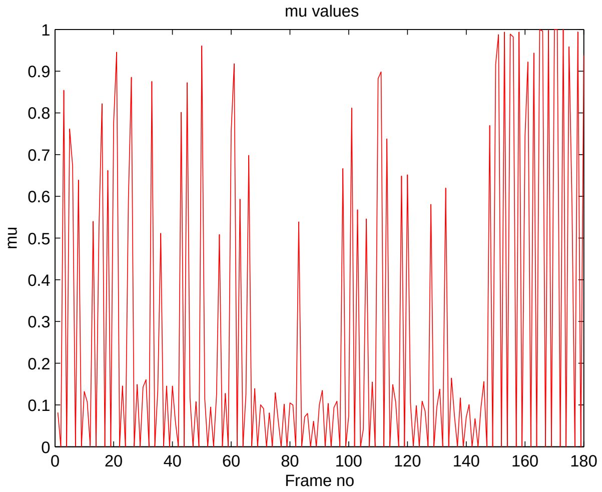
Twelve short videos are used to test the program that computes the jerkiness. The first six videos are of "avi" format and of QCIF resolution, while the remainder movies are of "YUV" format. It should be noted that in avi video format, dt is constant, i.e, the time interval for each frame display in the video frame sequence is not changing and causes a constant value of $\tau(t)$ throughout video display time. That is shown in (c) of all figures. Figures (4.1.1-4.1.12), show plots of the motion intensity, $\mu(m)$, and $\tau(t)$. Table (4.1) summarizes test cases details in addition to the resultant jerkiness values for each test case. First six (1- 6) videos are tested normally without any errors and three (7- 9) videos are tested before and after adding freeze to them. Figure 4.1.1, shows the results of "clip1.avi". It shows that this clip has high motion intensity for frames 150 and above, this leads to a $\mu(m)$ close to 1 for those frames as shown in (b). This clip is of avi format so $\triangle t$ is uniform, therefor $\tau(t)$ is constant. Figure 4.1.2, shows the results of "clip2.avi", it shows also that the number of frames is larger (˜1200). This clip is of avi format, $\triangle t$ is uniform, therefor $\tau(t)$ is constant but higher than $\tau(t)$ of "clip1.avi" because of higher number of frames. The jerkiness value of "clip2.avi" is higher than "clip1.avi" because it has larger number of frames, a significant portion of frames has high value of motion intensity and $\mu(m)$ close to 1. Figure 4.1.3, shows results of "clip3.avi", the number of frames is comparable to "clip1.avi" but the time step is larger so $\tau(t)$ is larger than of "clip1.avi". The jerkiness of clip3 is 50% larger than of clip1 due to higher value of $\tau(t)$ mainly. Figure 4.1.4, shows the results of "clip4.avi", the first 60 frames shows almost zero motion intensity and zero $\mu(m)$. Most of the frames after that shows a high value of motion intensity and a unit $\mu(m)$. The number of frames is larger than of "clip1", $\tau(t)$ is constant and of comparable value to clip1. The number of frames of almost unit $\mu(m)$ in clip4 is larger than of clip1 and all these factors make the jerkiness of clip4 higher than it was in clip1. Figure 4.1.(5-6), show results of “clip5.avi” and “clip6.avi”, there is a high value of motion intensity for the frames 180~340. For QCIF video, a high motion intensity means value above 5 and $\mu(m)$ is computed using the following second equation of the s-shaped function as described in chapter 3. $\mu(m)$ is almost unit for these frames because motion intensity is higher than 10.

$$\mu(m) = \frac{d}{1 + \exp(-c\,(m - pm))} + 1 - d$$

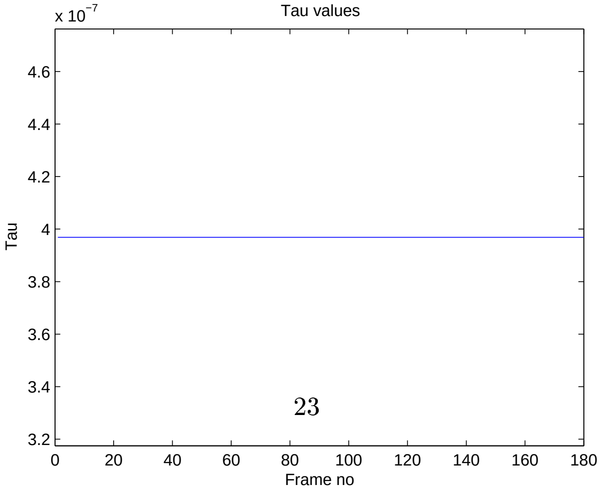
Figure 4.1.7, shows the results of “clip7.yuv”, it also shows that most of the frames have small value of motion intensity therefore a small value of $\mu(t)$. This video consists of 300 frames which is larger than the number of frames of clip1. The resulting jerkiness value of clip7 is smaller than of clip1 due to smaller motion intensities of the frames of clip7 compared to clip1 although clip7 has more frames than of clip1. Figure 4.1.8, shows results of “clip8.yuv”, this video consists of 150 frames which is half the number of frames of “clip7.yuv”. Same value of $\triangle t$ is observed for all “yuv” clips used in this study. Figures (4.1.8 b, 4.1.7 b), show that the number of frames with value higher than 5 in clip8 is larger than clip7 and this is why the jerkiness of “clip8.yuv” is larger than the jerkiness of “clip7.yuv”. Figure 4.1.9, shows the results of “clip9.yuv”, this video has 300 frames as “clip8.yuv” and has the same frame $\triangle t$. Comparing Figures (4.1.8 b, 4.1.9 b), we can see that clip9 has higher average $\mu(m)$ compared to clip8 and the result of that is a higher jerkiness value of clip9 compared to clip8. Figures 4.1.(10-11-12), show results of clips7/8/9 with freezing of 1/3 of its frames. Freezing frames are made by a freeze program that allows defining the freezing frames interval and replace the numerical data of all the frames in the freeze interval by the data of the first frame in that interval. The freezing procedure causes zero motion intensities of the frames in the freezing interval and this causes zero value of $\mu(m)$. The jerkiness value of the videos with intentional frame interval freeze is lower than the jerkiness value of the videos without freezing as shown in Table (4.1) when comparing case (7 and 10), (8 and 11), and (9 and 12).



(a) Motion Intensity

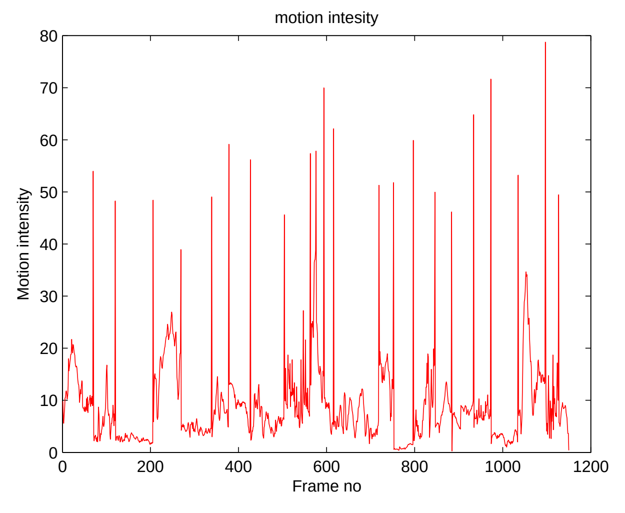


(b)$\mu(t)$

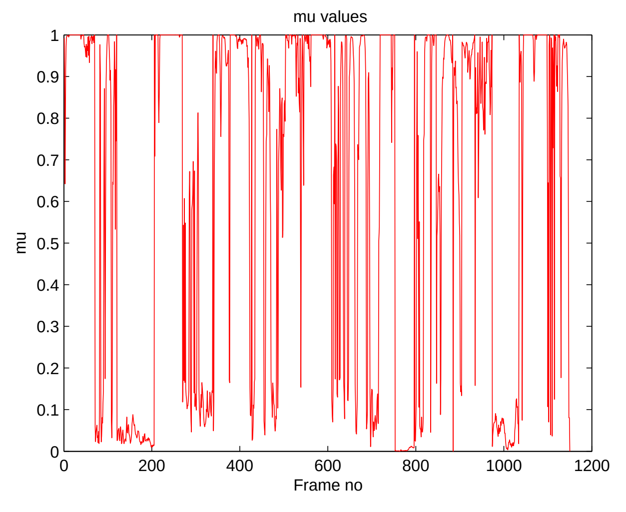


(c) $\tau(t)$

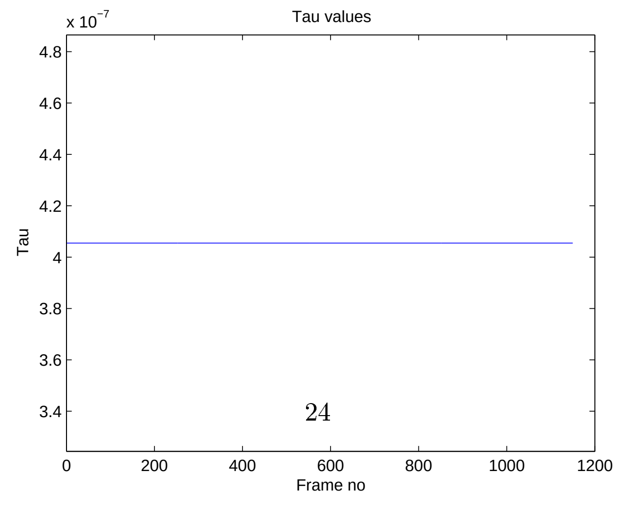
Figure 4.1.1: Test Case 1 Results



(a) Motion Intensity

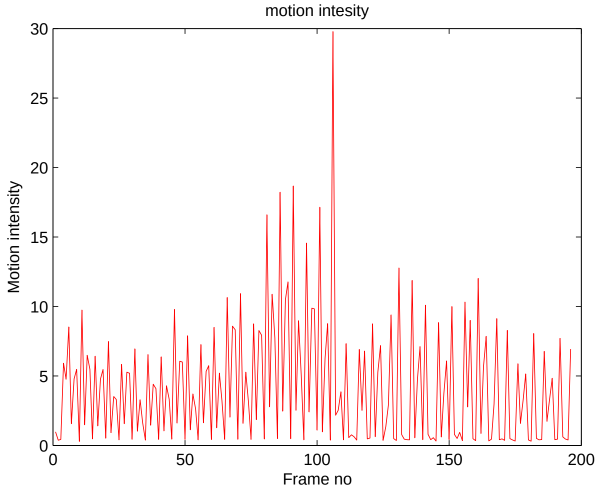


(b)$\mu(t)$

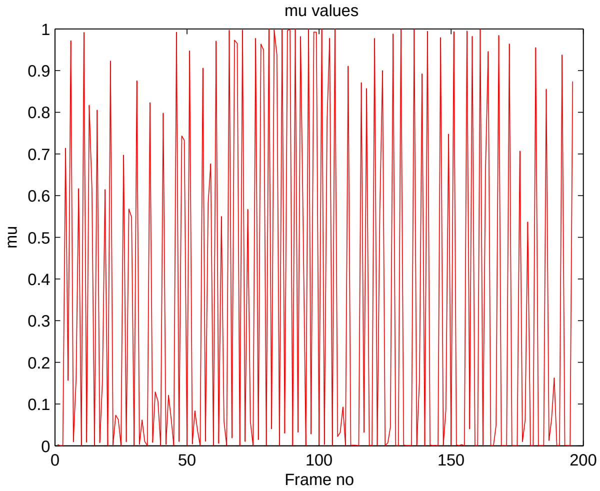


(c) $\tau(t)$

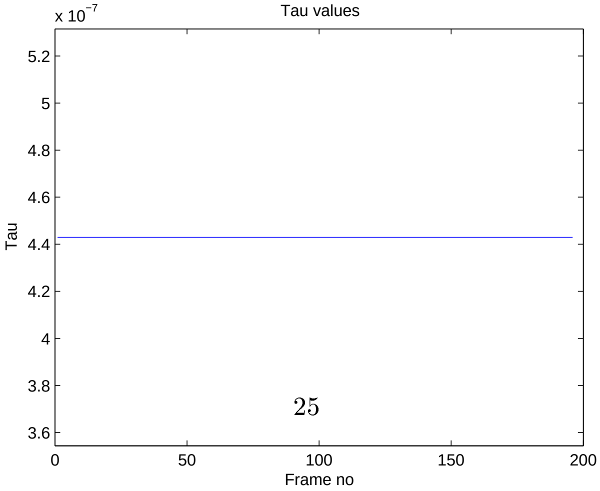
Figure 4.1.2: Test Case 2 Results



(a) Motion Intensity

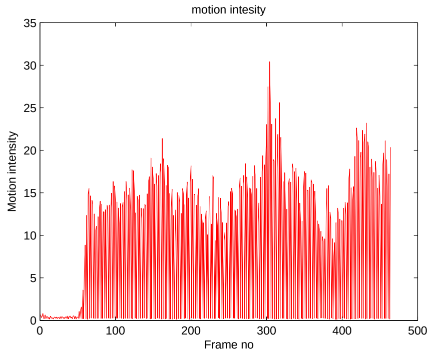


(b)$\mu(t)$

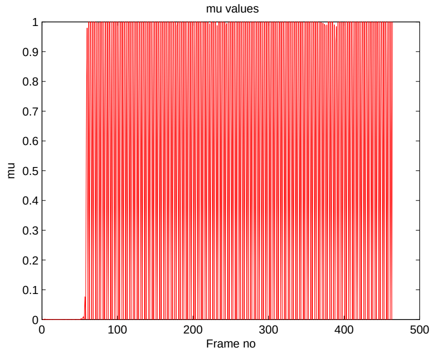


(c) $\tau(t)$

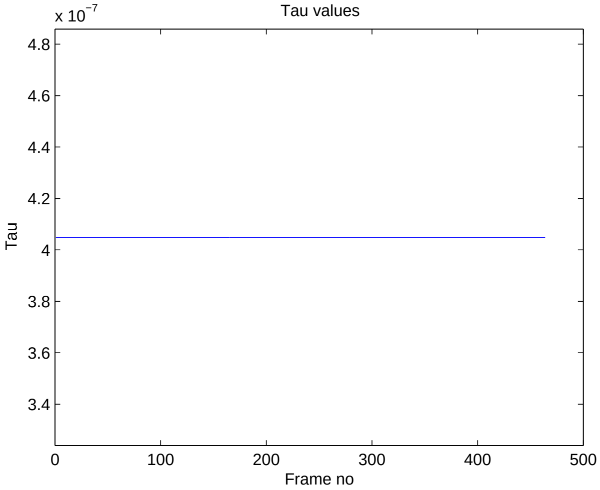
Figure 4.1.3: Test Case 3 Results



(a) Motion Intensity

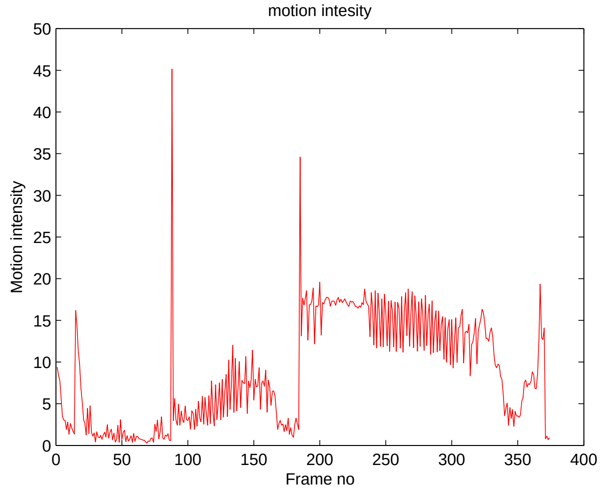


(b)$\mu(t)$

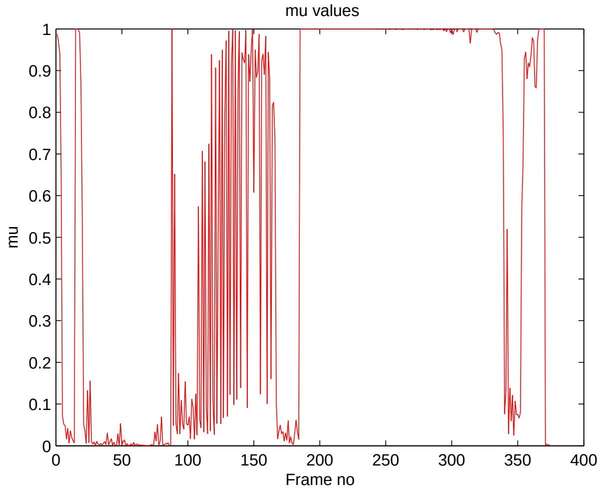


(c) $\tau(t)$

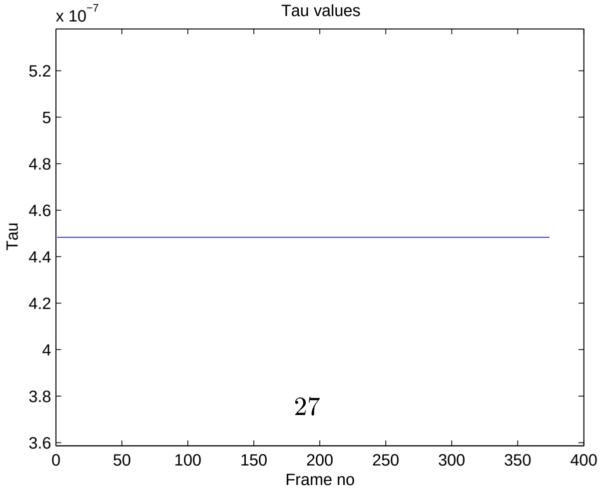
Figure 4.1.4: Test Case 4 Results



(a) Motion Intensity

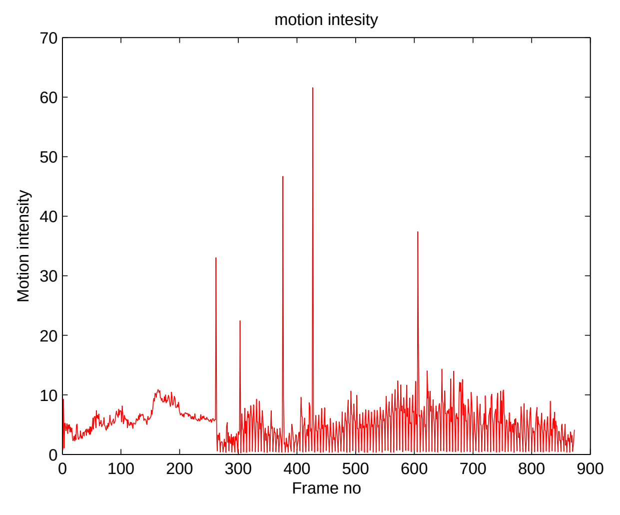


(b)$\mu(t)$

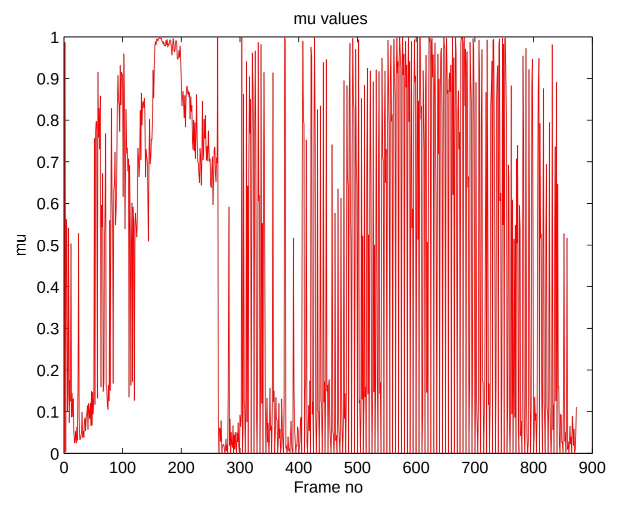


(c) $\tau(t)$

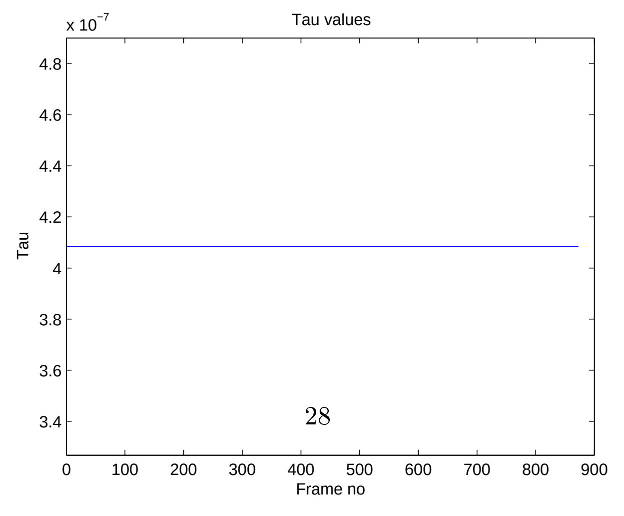
Figure 4.1.5: Test Case 5 Results



(a) Motion Intensity

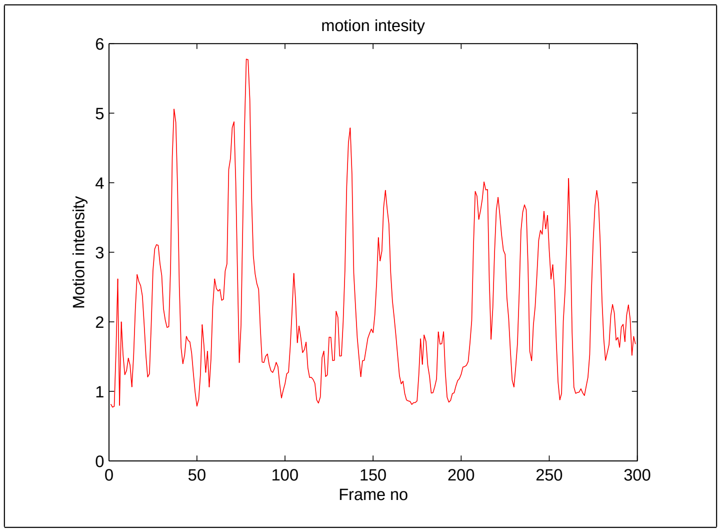


(b)$\mu(t)$

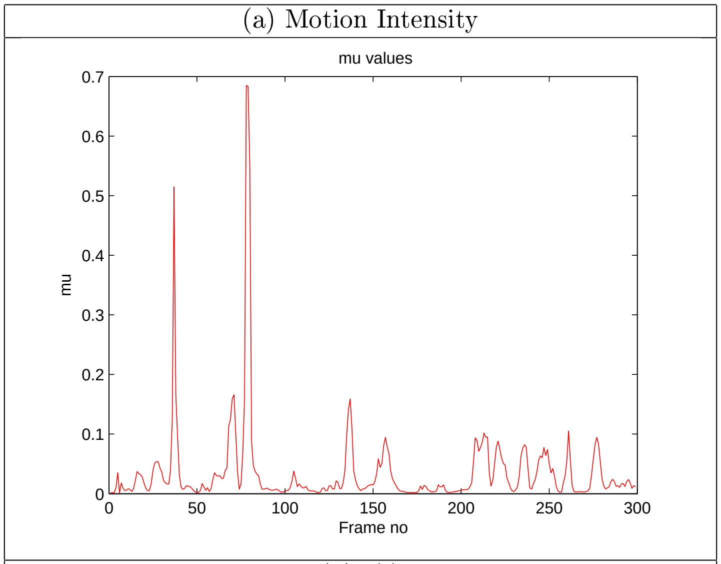


(c) $\tau(t)$

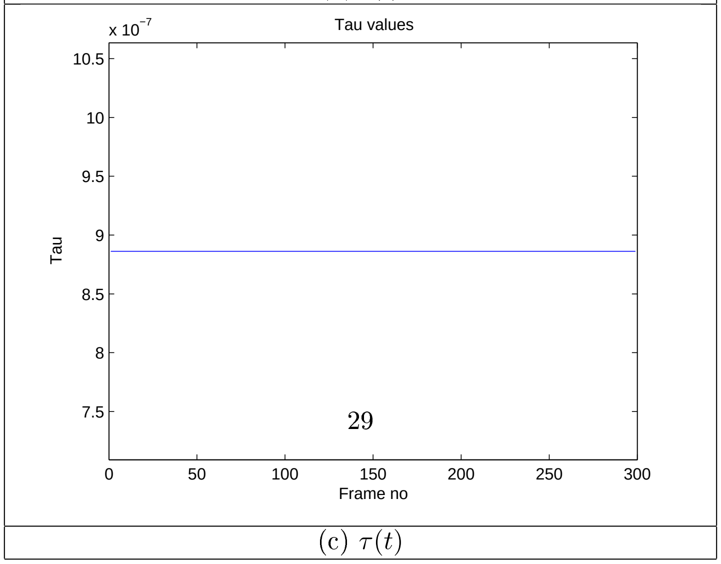
Figure 4.1.6: Test Case 6 Results



(a) Motion Intensity

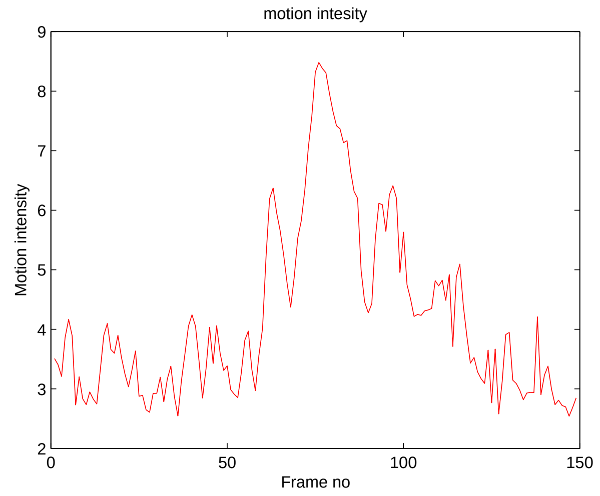


(b)$\mu(t)$

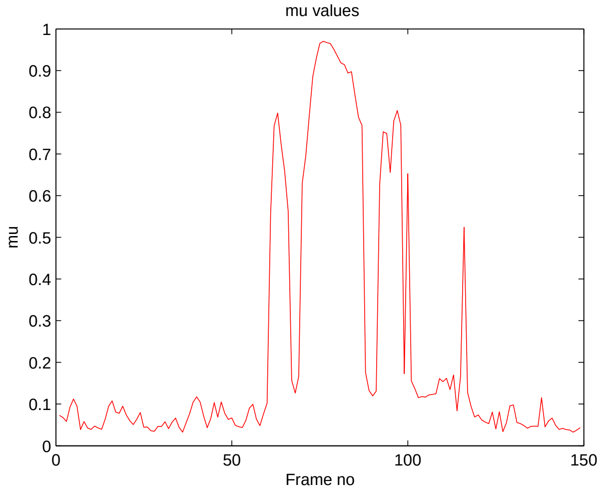


(c) $\tau(t)$

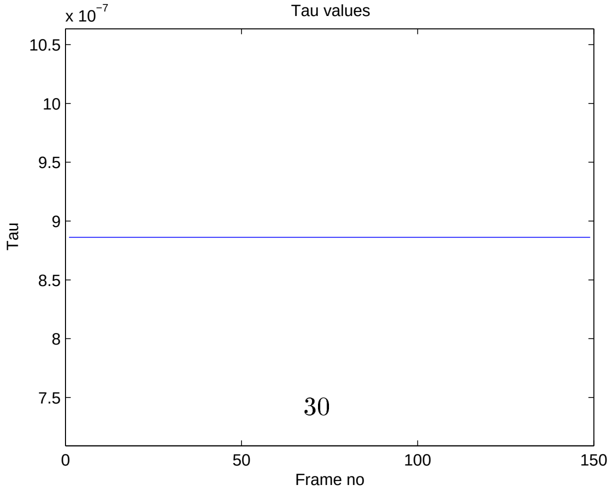
Figure 4.1.7: Test Case 7 Results



(a) Motion Intensity

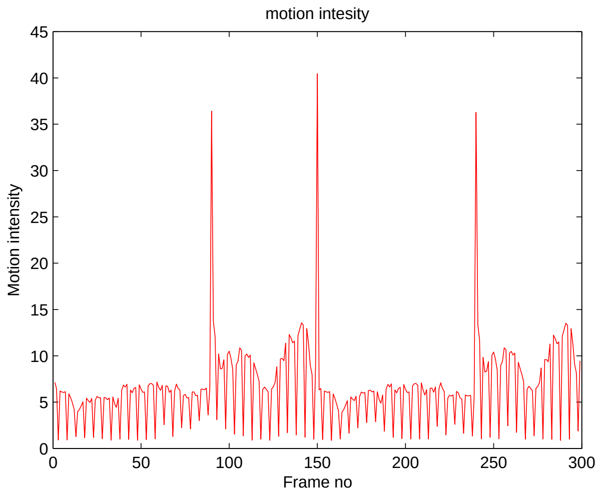


(b)$\mu(t)$

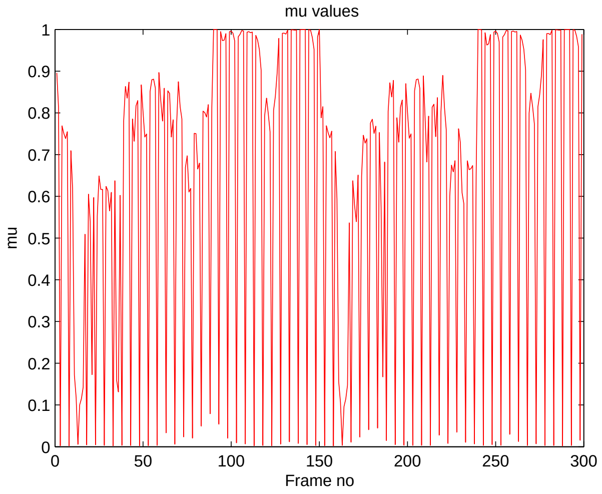


(c) $\tau(t)$

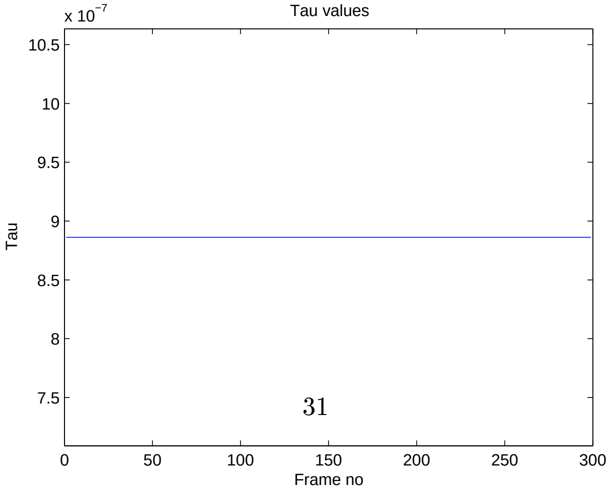
Figure 4.1.8: Test Case 8 Results



(a) Motion Intensity

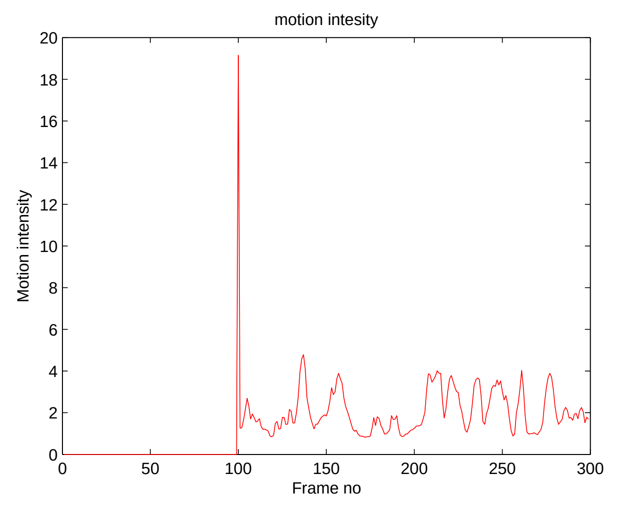


(b)$\mu(t)$

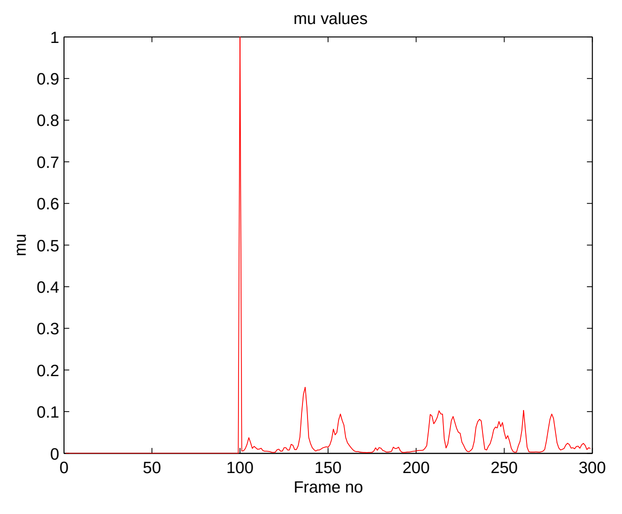


(c) $\tau(t)$

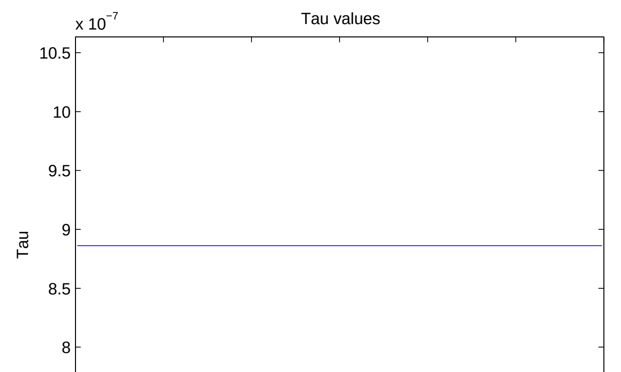
Figure 4.1.9: Test Case 9 Results



(a) Motion Intensity

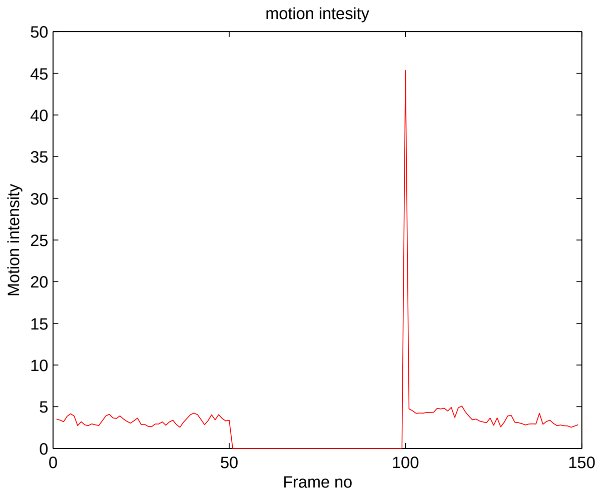


(b)$\mu(t)$

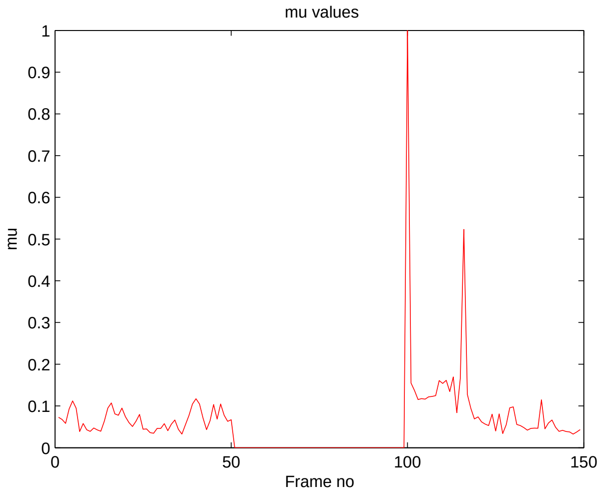


(c) $\tau(t)$

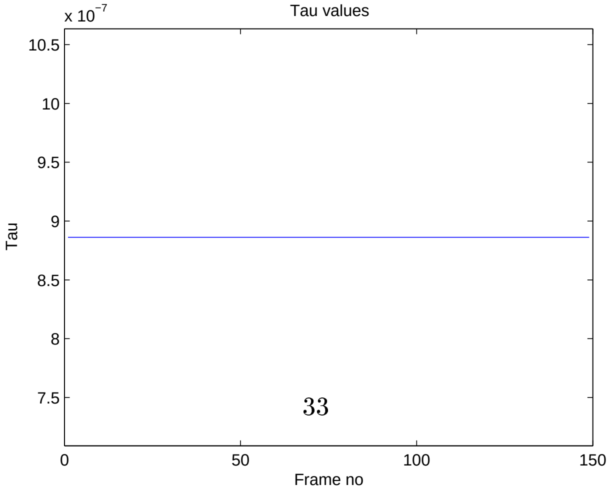
Figure 4.1.10: Test Case 7 with freeze Results



(a) Motion Intensity

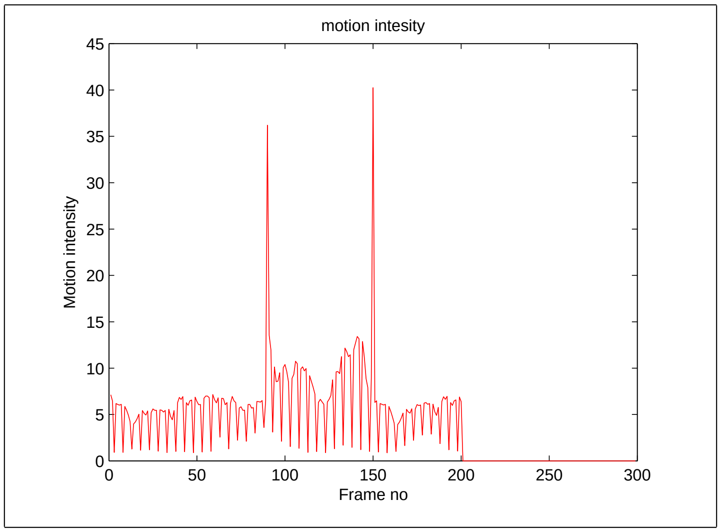


(b)$\mu(t)$

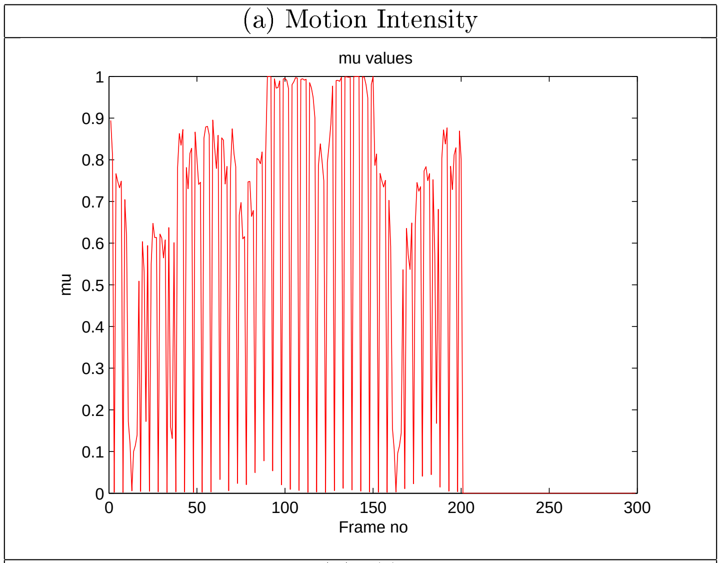


(c) $\tau(t)$

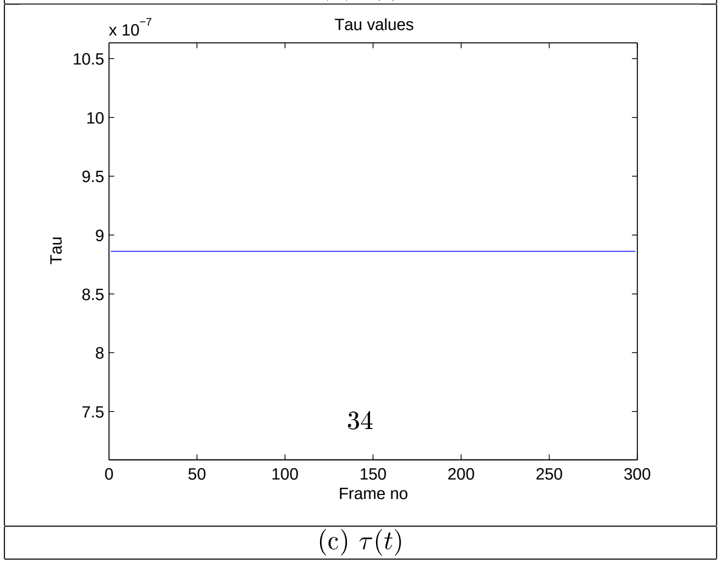
Figure 4.1.11: Test Case 8 with freeze Results



(a) Motion Intensity



(b) $\mu(t)$



(c) $\tau(t)$

Figure 4.1.12: Test Case 9 with freeze Results

case	No of Frames	$\triangle t$	Jerkiness
1	181	0.0400	1.0726 E-7
2	1157	0.0402	2.5963 E-7
3	197	0.0412	1.5205 E-7
4	465	0.0402	2.1209 E-7
5	375	0.0414	2.6523 E-7
6	874	0.0403	1.8876 E-7
7	300	0.05	3.1592 E-8
8	150	0.05	2.0844 E-7
9	300	0.05	5.6480 E-7
7+f	300	0.05	1.8923 E-8
8+f	150	0.05	5.2312 E-8
9+f	300	0.05	3.5826 E-7

Table 4.1: Summary of results

4.2 Conclusion

A MATLAB code is implemented to compute video jerkiness to help in video transmission quality assessment. Table(4.1) shows that the video jerkiness for videos without frame freeze (cases 7,8,9) is higher than the jerkiness value with frame freeze (cases 10,11,12) which is an indication of successful implementation of jerkiness computation procedure presented by Borer [3]. The motion intensity of the frames in the freezing interval causes zero value of $\mu(m)$ so by substituting motion intensity values in the following model equation some equations values are almost equal zero, and after summation the total value are less that the value of summation equations with more motion intensity values.

$$J\left(v\right)=\frac{1}{T}\sum_{\infty}\triangle t_i\cdot\tau_\alpha\left(\triangle t_i\right)\cdot\mu\left(m_{i+1}\left(v\right)\right) \tag{4.2.1}$$

It is proved also that all the YUV videos used have the same $\triangle t$ while each AVI video have its own constant $\triangle t$ through all the video duration. $\triangle t$ is shown to be directly proportional to τ as calculated in Sshape function as well. For QCIF video, a high motion intensity means value above 5 and $\mu(m)$ is computed using the following second equation of the s-shaped function as described in previous chapters. The $\mu(m)$ is almost unit (1) when motion intensity is higher than 10.

$$\mu(m)=\frac{d}{1+\exp(-c\,(m-pm))}+1-d$$

4.3 Future work

This thesis implements a program to measure Jerkiness in video samples of QCIF resolution. As a future work, this program can be extended to deal with other formats videos and higher resolutions than QCIF. The implemented program can be used for video transmission quality assessment via MATLAB, to compute the different jerkiness values and develop a database to help studying the Jerkiness values of different videos with different formats, frame rate, resolutions and $\triangle t$s.

It worth to be mentioned, that jerkiness calculation can be used as a quality measure of transmission. If the jerkiness value before and after transmission is calculated, i.e., $J_{before}$and J_{after}, then the percentage of quality of transmission can be calculated as the following:

$$Q_{transmission}=\frac{J_{after}}{J_{before}}\times 100\ \%$$

Example:

Assume the jerkiness before transmission is substituted by jerkiness value of a freeze free video (jerkiness value of test case 7 freeze free), while jerkiness after transmission is substituted by jerkiness value of the same video after adding freeze (jerkiness value of test case 7 with freeze).

$$Q_{transmission} = \frac{1.8923e-008}{3.1592e-008} \times 100 \ \%$$

$$Q_{transmission}{=}59.9 \ \%$$

Different programming languages can be used as well to develop similar algorithms to help reaching the same aim which is to deliver the best video quality to an end user who uses various communications systems.

Bibliography

[1] Gina De Angelis. *Motion Pictures.* The Oliver Press, 2003.

[2] K. Blair Benson. *Television Engineering Handbook.* McGraw-Hill, 1986.

[3] S. Borer. A model of jerkiness for temporal impairments in video transmission. pages 218–223, 2010.

[4] Mohammed Ghanbari and Quan Huynh-Thu. No-reference temporal quality metric for video impaired by frame freeze artefacts. 2010.

[5] VQEG HDTV Group. VQEG HDTV Final Report version2.0 video quality models. volume 2.5, 2008. http://www.its.bldrdoc.gov/vqeg/projects.aspx.

[6] W. Wharton & D. Howorth. *Principles of Television Reception.* Pitman Publishing, 1971.

[7] Anil K. Jain. *Fundamentals of digital image processing.* Prentice Hall, 1989.

[8] Ricardo R. Pastrana-Vidal and Jean-Charles Gicquel. Automatic quality assessment of video fluidity using a no-reference metric. 2007.

[9] Ricardo R. Pastrana-Vidal and Jean-Charles Gicquel. A no-reference video quality metric based on a human assessment model. 2009.

[10] Charles A. Poynton. *Digital Video and HDTV: Algorithms and Interfaces.* Morgan Kaufmann, 2003.

[11] Dima Pröfrock. Yuv to matlab movie. MATLAB, 2011. http://www.mathworks.se/matlabcentral/fileexchange/authors/24073.

[12] S. Winkler. *Digital Video Quality: Vision Models and Metrics.* John Wiley and Sons, 2005.