Effect of Temporal Filters on Face Images

Rasheed Rebar Ihsan

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Eastern Mediterranean University
January 2017
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Mustafa Tümer
Director

I certify that this thesis satisfies the requirements as thesis for the degree of Master of Science in Computer Engineering.

Prof. Dr. Işık Aybay
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Computer Engineering.

Assoc. Prof. Dr. Mehmet Bodur
Supervisor

Examining Committee

1. Assoc. Prof. Dr. Mehmet Bodur
2. Asst. Prof. Dr. Adnan Acan
3. Asst. Prof. Dr. Ahmet Ünveren

ABSTRACT

Face detection from low-resolution videos is a challenging research area. This thesis explores the effect of a temporal filtering method by Dr. Bodur on face images. The temporal mean and median filters calculate the intensity of pixels using the intensities of surrounding neighbour pixels, and temporal neighbour pixels in consecutive images. The effect of the proposed technique on the image is measured by the mean square error (MSE) and the peak signal noise ratio (PSNR) values using the pixels of the original high resolution image as reference values to measure the error and noise figures of the pixels of filtered low resolution images. Results demonstrate a significant effect of the proposed filters on the consecutıve frames of face vıdeo record. In the tests, the medıan fılter ıs found more effective compared to mean fılter.

Keywords: Temporal Mean Filter, Temporal Median Filter, Image Resolution, Effectiveness of Image Filter.

ÖZ

Düşük çözünürlüklü videolardan yüz tanıma zorlu bir araştırma alanıdır. Bu tezde, Dr. M. Bodur'un önerisi olan zaman boyutlu filtreleme yönteminin yüz görüntüleri üzerindeki etkisi incelenmektedir. Zamansal ortalama, medyan ve maksimum filtrelerde her pikselin parlaklığı, çevreleyen komşu piksellerin yanısıra ardışık görüntülerdeki zamansal komşu piksellerin yoğunlukları da kullanarak hesaplanır. Önerilen tekniğin görüntü üzerindeki etkisi, düşük çözünürlüklü görüntülerin piksellerini referans amaçlı kullanılan orijinal yüksek çözünürlüklü görüntünün pikselleriyle karşılaştırarak ortalama karesel hata (MSE) ve tepe sinyal gürültü oranı (PSNR) olarak elde edilmiştir. Testlerde, ortalama fıltre ile karşılaştırıldığında medyan fıltrenin daha etkin olduğu görülmüştür.

Anahtar Kelimeler: Zamanda Ortalama Filtre, Zamanda Medyan Filtre, Görüntü Çözünürlüğü, Görüntü Filtre Etkinliği.

DEDICATION

To my *parents* who helped me to be stronger and better person

To my lovely *sisters* and *brothers*

To everyone who encouraged me

ACKNOWLEDGMENT

In the present world of competition there is a race of existence in which those are having will to come forward succeed. Project is like a bridge between theoretical and practical working. With this willing I joined this particular project.

First of all, I would like to thank the supreme power the Almighty God who is obviously the one has always guided me to work on the one has always guided me to work on the right path of life. Without his grace this project could not become a reality. Next to him are my parents, whom I am greatly indebted for me, brought up with love and encouragement to this stage.

I sincerely thank to my supervisor Assoc. Prof. Dr. Mehmet Bodur, for his patience, motivation, enthusiasm, and knowledge. His guidance helped me in all the time of research and writing of this thesis. I thank my external jury members, Assist. Prof. Dr. Adnan Acan, and Assist. Prof. Dr. Ahmet Ünveren for their reviews and guidance. And special thanks to my worthy teacher of English, Assist. Prof. Dr. Nilgun Hancioglu for her helping and encouraging. Moreover, I sincerely thank to all the staff members of computer engineering department for their generous attitude and friendly behaviour. At last but not the least I am thankful to all my teachers and friends especially Pawan and Diler who have been always helping and encouraging me though out the year. I have no valuable words to express my thanks, but my heart is still full of the favours received from every person.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Fundamental Information

Face detection technology is extensively used in our daily life, especially in the area of real-time monitor, video tracking, criminal inspection, and etc. It is an easy work for humans to detect and recognize human faces by eyes; however, it is not an easy case for computers to do that. Face detection technology has been significantly developed, but a number of challenges waiting for solution. 1) A wide diversity of face models make the limited sample set very difficult to cover all the faces, and establish accurate distribution model in the high-dimensional space. 2) Human faces and optical conditions the background areas that are similar to human face. 3) The present face detection algorithms cannot cover arbitrary pose, lighting condition or faces with invisible parts. 4) Real-time detection system. Face detection in video has the requirement of real-time detection [1].

Recently face detection and face recognition had gotten significant attention from scholars in biometrics, computer vision communities, and pattern recognition. As should be obvious, both methods frameworks are essential in our day by day life [3].

The followings are challenges related with to face detection systems [7]:

- Variations in illumination: When the picture is shaped, factors such as lighting (intensity, source distribution and spectra) and camera features (lenses and sensor response) affect to some degree the appearance of the human face. Figure 1.2 shows the Illumination variations.

Figure 1.1: Example of variation in illumination
with permission by Ali Tarhini, 2010 [14]

- Pose variations: The pictures of a face differ as a result of the relative camera face pose (frontal, 45°, 90°, topsy-turvy, and some features of face for example the nose or eyes may become partially or completely occluded. Figure 1.3 illustrates the changes appearance due to pose variation.

Figure 1.2: Example of pose variation with permission by F.Tarrés *"GTAV Face Database"* [15]

- Expressions variation/facial style: The presence of faces is straightforwardly influenced by an individual's facial expression as illustrated in Figure 1.4. Facial hair, for example, moustache and beard can change facial appearance and characteristics in the lower half of the face, particularly close to the mouth and bottom areas.

Figure 1.3: Variation in appearance of an individual due to expression [16]

- Occlusion: Faces might be partially blocked by other objects. In a picture with a set of individuals, a few of the faces or other items may partly occlude other faces, which thus brings about just a little part of the face are accessible in several situations. Examples of partially occluded faces are illustrated in Figure 1.5.

Figure 1.4: Example of partially occluded faces with permission by A.M. Martinez, the AR Face Database [17]

1.2 Evaluation of Effects of Filters on Pixels of Image

The following are used as measure of effect of image filters on face images:

Mean Square Error (MSE) is considered to be an important criterion for evaluating the differences of between the filtered and non-filtered images. It is used as part of the digital image processing technique to measure the differences between the predicted and target images. It is well known that the main difference between estimators and predictors is constants are estimated and random variables are predicted. Additionally, MSE can also be used for passing on the concepts of bias, accuracy and precision in statistical estimation. MSE can be examined by knowing the target of prediction or estimation, and an estimator or predictor which is a data function [2].The formula and method of computation are described in more details in chapter 2.

The peak signal-to-noise ratio, sometimes shortened as PSNR, is one of the engineering terms which show the ratio among the signal's noise power and maximum power of a signal’s. PNSR is widely used by Engineers for measuring the performance of reconstructed pictures which have been compressed. There is a colour value for each of the image element (pixel). When a picture is compressed and then uncompressed, the colour of the picture changes. PSNR is found to be expressed in terms of the logarithmic decibel scale since signals might have an extensive dynamic range [4]. The formula and method of computation are described in more details in chapter 2.

1.3 Spatial filters for Improving the Face Image

Median Filter is a nonlinear smoothing approach that decreases the edges blurring and the current point in the image to be replaced by the median of the brightness in its neighbourhood is the idea behind it. As for personal noise spikes, these do not influence the median of the radiance in the neighbourhood and so median smoothing eliminates impulse noise in an outstanding manner [19]. **Mean Filter** the mean filter's function is to replace every pixel by the mean value of the intensities in its region. It can locally decrease the variation and can be implemented easily. It has the ability to smooth and blur the image and for additive Gaussian noise in the meaning of mean square error is excellent. Dappled image is that image in which multiplicative model is added with non-Gaussian noise and subsequently, the simple mean filter does not function well in this situation [18].

1.4 Surveillance Application for Face Detection

Surveillance is oversight of behaviours to obtain information. This definition contains a huge number of methods and mechanisms which can be deemed a format of surveillance. Several of these are identifiable out of public knowledge generated by popular civilization. The well known strategies for fixed surveillance systems are

(a) technical monitoring (commonly secret video recording or voice recording), and

(b) electronic monitoring (digital surveillance, counting of keystroke), and several more [8]. Surveillance is a valuable tool for the governments to observe and identify the people, threats, and criminal action [9].

Face detection surveillance is a part of surveillance which relies on features face for locating and recognition of human by artificial intelligence methods [10]. With increasing demands for protection and security of regions and belongings, nowadays biometric surveillance is a crucial system. Face is one of the most widely use of biometric feature. Observation pictures and recordings caught utilizing different sensors are principle hotspots for reporting and documenting the checked exercises of concern [3]. In many environs observation video systems are shared and widespread. Video observation has been a key component to accomplish safety at banks, correctional institutions, airports, and gaming club [11]. Some of the frames illustrated in Figure 1.6 that shows humans recorded by surveillance video.

Figure 1.5: Set of frames from videos surveillance with permission by Cisco Physical Security, 2014 [12]

1.4 Problem Statement

In many applications related to security a low quality video shall be processed to determine the features of faces in the video. The accuracy of extracting the facial features is a critical step of identification of the persons in the captured videos.

For a poor quality video with *n* frames $\{ f_1, \mathrm{f}_2, \mathrm{f}_3, \dots, \mathrm{f}_\mathrm{n}\}$ using each frame to find the face features brings considerable loss of information, and high errors of face features. A sequence of frames is expected to have additional facial information for accurate detection of faces. This thesis tests the effectiveness of a set of temporal filters developed by the Supervisor of this thesis, Dr. Mehmet Bodur. Temporary filters combine the information of the consequent image frames into a single image to decrease the error in determination of face images.

To get the effect of filtering, an experimental procedure is developed based on the measured difference of the filtered and the raw images for 50 facial videos. The proposed experimentation starts with high quality images which is expected to provide best face recognition results. HD videos were processed to reduce their size in 10% steps so that a set of lower and lower quality images are obtained from the initial HD images. The filtered and non-filtered low quality images are compared to determine the effect of the filters on the image pixels.

Detecting human face in video is a hard issue due to the existence of large differences in facial pose and lights, and poor quality of image. Many image processing methods were proposed in literature but non of them measures the effect of the filters on the pixels of the images. This thesis compares the effects of standard mean and median filters to the effects of temporal mean and median filters.

Chapter 2

LITERATURE SURVEY ON FACE DETECTION

2.1 Categories of Face Detection Strategies

Face detection is one of the main problems in the field image processing and computer vision. Many security applications recently used face detection, and collected intensive attention. In the literature, face detection strategies can be separated in three categories, based on the data acquisition methodology of face images: (a) the approaches that work on the intensity of the images, (b) those which need some sensory data like *3D* information or infra-red metaphors and (c) with video sequences [22]. Detecting face in the image is accomplished by the method called Viola-jons, which calculates the features to detect the exact region of the face.

In this chapter an overview is presented on different approaches by several researchers, common features of video recording, and image filtering methods with two spatial filters, interpolation, face detection, performance evaluation and the chapter concludes with a summary.

2.2 Common Approaches of Face Detection Methods

There are a number of methods available for the identification of a person face. This section covers the discussion on a number of approaches for facial detection, such as face localization, facial feature detection, face tracking and colour segmentation techniques.

Face localization: means to decide the picture location of a solitary face; this is an abridged detection issue with the supposition that an input image contains one and only one face.

Facial feature detection: The objective of facial characteristic detection is to detect the presence and position of characteristic, for example nose, eyes, lips, eyebrow, nostrils, ears, mouth, etc., with the supposition that there is a single face in a picture.

Face tracking: the technique of continuously guessing the position and possibly the direction of a face in a picture series in real time [5].

Colour segmentation techniques: This method utilizes the skin colour to separate the face. The areas which comprise non-skin colour regions on the face are viewed as candidates for eyes and/or mouth. Therefore, such techniques' performances on facial image databases are to a certain degree inadequate. This is because of ethnical backgrounds of different people [20].

2.3 Common Image Representation Methods

Video is the term used for the recording, reproducing, or broadcasting of moving visual image. Just as other media extensions differ in terms of resolution, so does video systems. For development of algorithms on a video record, it is essential to understand the formats of a video. The present study used MP4 as the standard format for the target face videos. A video is made of a set of consequent images, which are reflected on a display at periodic time instants.

Number of frames per second: The number of the still images per the unit of video's duration is known as frame rate and it generally ranges from six to eight frames per second for traditional cameras and for new cameras it reaches up to more than 120 frames per second. The minimum number of frames that can produce the stimulation of a moving picture is around sixteen frames per second. Each frame consists of the pixels of an image. The video used in this thesis has been recorded at 29 frames per second.

Format of an image: A two dimensional function is known as an image in which f is (*x*, *y*), x in this case and y are spatial coordinates, the intensity is described as the amplitude of coordinate pairs (*x*, *y*) of the points on an image. **Pixel** is the smallest piece of an image. Each pixel corresponds to any one value. The value of a pixel at any point corresponds to the intensity of the light photons striking at that point. Each pixel stores a value proportional to the light intensity at that particular location. In each picture, there may be thousands of pixels that together make up an image. There are many types of images with the colour distribution in them: (a) **The binary image:** The binary, or monochrome image, as suggested by the name, possesses solely two pixels in which the both pixels refer to white and black colour; (b) **8 bit colour format:** Also called grayscale image, each pixel is represented by 8 bit, which corresponds to 256 varying shades of colours; (c) **24 bit colour format:** The true colour format, also known as 24 bit, is allocated in three extensions, i.e. red, green and blue extensions. This is because 24 can be equally divided on 8 and into three different channels of colours.

For the purpose of object detection, the colours used in the image provide the basis. It also helps for image tracking and recognition and so on. For this reason, the 24 bit RGB colour was used in this research.

The image files that are most commonly used in cameras, printers, scanners, internets and so on are the JPG, TIF, PNG, and GIF.

2.4 Mean Filter

Mean filter is described as a simple method which can be implemented easily for smoothing out images and to suppress the noise by reducing the intensity variation among the pixels. The whole notion works on replacing the values of pixels in different images with those of its neighbours. Consequently it reduces the effect of pixels which are extremely different than their neighbours. The mean filter is also considered as a form convolution filter. A kernel surrounds the target pixel indicating the effect of the neighbouring pixels. Mean filters mostly use a 3x3 square kernel, as seen in Figure 2.2. Other sizes of square kernels such as 5x5 are used especially for extreme smoothing. Reducing noise of an image is one of the advantages of a mean filter. And, it has the disadvantage of losing details and turning the image blurry [19].

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Figure 2.1: 3×3 Averaging kernel used in mean filtering

The average (mean) is defined as:

$$\hat{f}(x, y) = \frac{1}{m \times n} \sum_{(s,t)\in Sxy} g(s,t)\ , \qquad (2.1)$$

Where *g(x, y)* is the original image, on the other hand *g(s, t)* is the sub-image whose dimension are *m x n*, $\hat{f}(x, y)$ is the image that is filtered. As for the sub-images, they are, after being summed up, divided by *m x n*, and the *x, y* stand for the dimensions of the image, and therefore, $s_{xy}$is the sum of all the pixels in a 3x3 region (m=n=3) and (s, t) is a pixel which belongs to the s_{xy} set.

5	3	6
2	$\boxed{1}$	9
8	4	7

Mean Filter
mean= (5+3+6+2+1+9+8+4+7)/9= 5
45/9

*	*	*
*	$\boxed{5}$	*
*	*	*

Figure 2.2: Intensity calculation of 3x3 mean filter

The value of the centre at the beginning (1) by applying mean replaces to (5).

2.5 Median Filter

Similar to the mean filter, the median filter is commonly used for reducing image noise. The median filter is far better than the mean filter for keeping the details in an image.

Similar to the mean filter, the median filter operates on each pixel in a picture, to replace the pixel value by a better representative, the median value, of the surrounding pixels.

The median value is defined the value at the mid of the sorted list of all entries. It is calculated by sorting out the values of the kernel pixels including the surrounding neighbourhood, and selecting the value at the middle of the sorted list to replace the target pixel.

5	10	2
2	$\boxed{1}$	3
8	4	7

Median Filter
sorted to 1,2,2,3,**4**,5,7,8,9. Take the number at the middle: 4

*	*	*
*	$\boxed{4}$	*
*	*	*

Figure 2.3: Demonstration of 3x3 median filter

The median filter has two main advantages over the mean filter over mean filter: (1) Unrepresentative neighbouring pixels will not affect the median value because of the median's average while it influence the mean. (2) The median filter does not produce unrealistic pixel values since the median value is one of the neighbouring pixels. This is especially important at the edges of the image when half of the kernel is empty [18].

2.6 Resizing of an Image by Interpolation

Interpolation is the estimation of intensities of pixel when we enlarge an image or reduce the size of image the output image contains more pixels or fewer pixels than the original image. A survey of interpolation systems in the medical image processing domain is presented in [6] it shows in detail how the interpolation work. Mainly interpolation is carried by non−adaptive, and adaptive techniques. Some of the non-adaptive systems are: bi-cubic interpolation, bi-linear and nearest neighbour. Non-adaptive techniques are attractive and are widely used due to their ease of computation. Adaptive techniques rely on the intrinsic features of an image such hue, edge information, etc. Alternatively, non-adaptive techniques do not utilize any intrinsic feature of an image and apply a specific computational logic on the intensities of image pixel to interpolate an image.

2.7 Face Detection

The face detection in an image is a material that has always been surveyed in computer literature vision. The face detection algorithm is accorded to the diversities in illumination, visual angle, background and facial expressions and the executing are not easy. Face detection can be utilized in lots applications such as video surveillance; face recognition, driver face observation and or human computer interface, image database management; human face detection is essential and hard process. Face detection algorithms are either based on (i) features or (ii) learning methods.

Viola – Jones algorithm is purposed for real – time faces detection from an image. Haar type features, computed rapidly by using integral images, feature selection using the AdaBoost algorithm (Adaptive Boost) and face detection with attentional cascade can be used by obtaining its real-time performance.

Viola-jones algorithm detects face robustly**,** having a very high rate of detection (true-positive rate) and with a very low rate of false-positive; in real time**,** approximately 2 frames per second providing only face detection**,** which means to differentiate faces from non-faces. Detection is mostly the first operation in the process of face recognition [4].

Integral image is a concept developed by Frank Crow. A part of the image bounded by corner points {a, b, c, d} is called integral image when it is subject to generate the sum of values in that grid [4].

Haar-masks are mostly used for calculating features. Starting from the mutual features of the faces such as the area around the eyes is much darker than the cheeks or the nose zone is brighter than the area of the eyes. Five Haar-masks (Figure 2.5) have been chosen for defining the features as calculated at various positions and sizes. Haar features are figured as the variation between the sum of the pixels from the white and black zones. In this way, it is somehow hard to expose contrast differences [4].

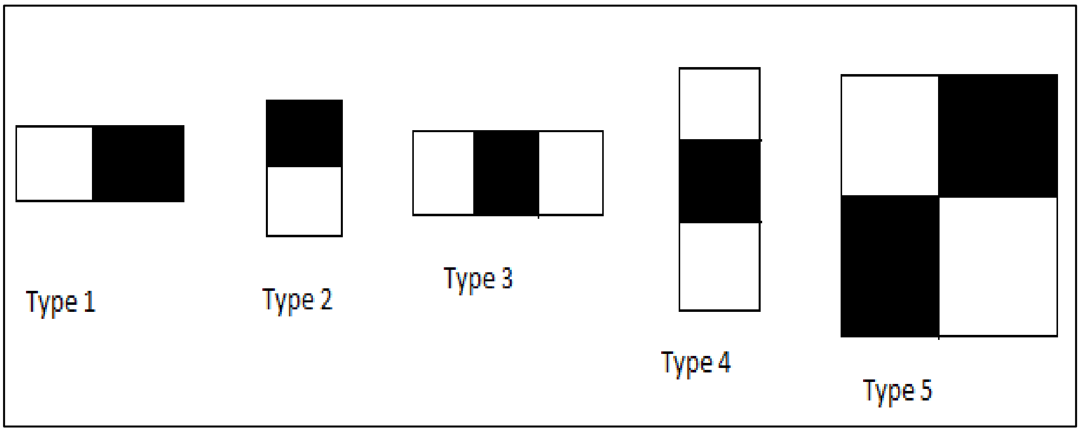

Figure 2.4: Types of features Haar-masks

Figure 2.5: Haar Feature, the intensity of pixels difference between eyes region and cheek region.

If we consider the mask M from Figure 2.6, the image is associated with the Haar-feature-I behind the mask is being defined by:

$$\sum_{1\leq i\leq N}\sum_{1\leq j\leq N} I(i, j)_{white} - I(i, j)_{black} \tag{2.2}$$

The Haar-feature-I behind the mask is defined by the integral image concept as the difference of pixel values in the white and black regions of the mask.

The features are taken out for windows with their dimensions of 24*24 pixels that have moved on the image where we like to detect faces. For such a window, Haar-masks are scaled and moved performing 162,336 of features. To decrease the Haar-features computation time, which are vary relying on the type and the size of the feature, the integral image was being used.

$$II\,(i, j) = \sum_{1\leq s\leq i} \sum_{1\leq t\leq j} I(s,t),\ \ 1 \leq i \leq N\,,1 \leq j \leq N \qquad (2.3)$$

Let A, B, C and D take the values of the integral image at the rectangle corners as seen in Figure 2.7. Then the total original image values in rectangle can be calculated: sum = $A - B - C + D$. Just 3 additions are necessary for any rectangle size. It is utilized in many areas of computer vision [4].

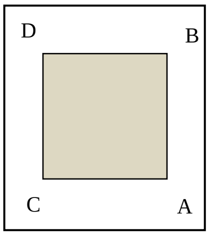

Figure 2.6: Sample of taking the values of integral image

2.8 AdaBoost Algorithm for Feature Selection

According to Haar-features for an image by 24*24 pixels can be as d = 162336, and most of them are excessive; AdaBoost algorithm had been used for a selection of a minimal number of features. The main idea is just to make an intricate classifier (decision rule) by utilizing a weighted linear by integrating of weak classifiers. Each feature is considered a weak classifier, as known by:

$$h\,(x, f, p, \theta) = \begin{cases} 1, \text{ if } pf(x) < p\theta \\ 0, \text{ otherwise} \end{cases} \qquad (2.4)$$

Where x is a 24 x 24 pixel image, Θ is a threshold and p is a parity. AdaBoost algorithm is founded on a set of training that includes n pairs (x_i,y_i), where x_i can be considered as positive or negative image. And y_i can be a label allocated to every single image and is equal to 1 for an image of positive and to -1 for an image of negative.

Attentional Cascade: After AdaBoost algorithm, a strong classifier will result that classifies the windows of *NxN* size well enough. Since, on average, only 0.01% of the windows are positive images, meaning faces, only potentially positive windows must be examined. Instead, to achieve a higher detection rate and a smaller misclassified images detection rate, we should use another strong classifier that classifies correctly the before misclassified images. This creates the attentional cascade, as showed in Figure 2.7. At the first layer of the attentional cascade, a strong classifier with few features is used, which will filter/reject most negative windows. A cascade of classifiers that are becoming more and more complex (with more features) will follow and they will allow to achieve a better detection rate. At each layer of the cascade, the negative images classified correctly will be eliminated and the new strong classifier will have a more difficult task than the previous step classifier

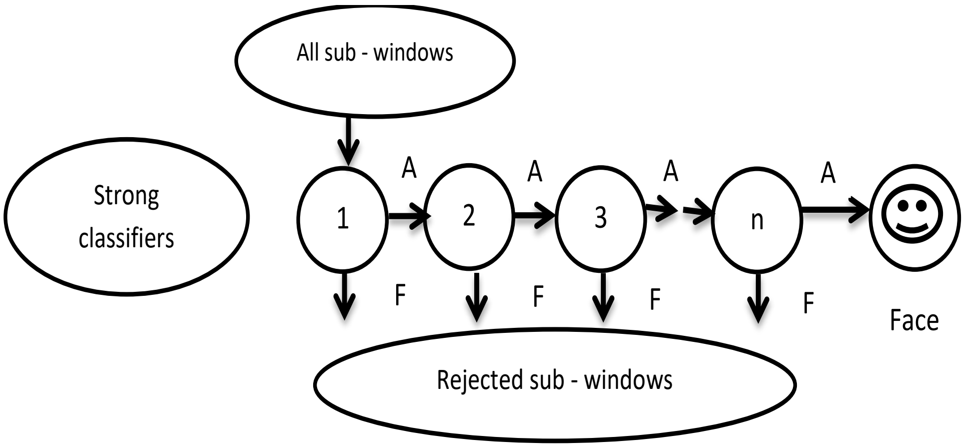

Figure 2.7: Attentional Cascade

Eventually, the cascade of classifiers will work as below:

The image will be divided into doubled windows; each window is an input inside the attentional cascade; at each layer, the window is being checked as it includes a face or not – due to the powerful classifier; if it is negative, the window is being declined and the step will be reiterated for another window; if it is positive, which means that window is a potential face and will go to the next layer of the cascade; the window includes of a face if it passes all attentional cascade layers [4].

2.9 Face Detector Implementation Using Matlab

Computer vision toolbox in Matlab contains cascade object detector (vision.CascadeObjectDetector) that makes a system object detector that is capable of detecting objects by using Viola – Jones algorithm. By default, the detector is a set to detect faces in an image, but it can also detect the mouth, nose, eyes or the upper part of the body known by the input string MODEL (ClassificationModel).

System object (detector) that detects faces from the image, by utilizing the Viola-Jones algorithm, the following common is used as such:

detector=vision.CascadeObjectDetector ('attentionalCascade.xml'), where the parameter is only performed under the name of xml file in which the attentional cascade was saved. After making of detector, the method pace is so-called by the next syntax: BBOX = pace (detector, I) that returns BBOX, an M- by – 4 matrix determining M bounding box including the detected objects. Each row includes of 4 components [x y width height] that assigns in pixels, the bounding box size and upper-left corner. By utilizing a detector acquired from training, the next paces are finished: (1) open the proper image; (2) make the detector object; (3) distinguish faces from the images; (4) add notes to the faces; (5) display image with adding notes to the faces [4].

2.10 Evaluation of the Effect of Filters on Pixels

The visual quality of an image can be improved or enhanced; nevertheless, the whole process is a subjective one. This is because no two persons can agree on the same image enhancement method. It, therefore, necessitates the use of empirical data to identify the effect of algorithms which are used for image quality enhancement.

There are two measures for evaluating in this thesis:

Mean square error (MSE) scores the difference for the pixels between two images, the original image and the produced image. MSE is calculated by:

$$\mathrm{MSE} = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} ||f(i,j) - \mathrm{g}(i,j)||^2 \qquad (2.5)$$

Where *f* stands for the matrix data of the original image; *g* stands for the matrix data of the degraded image to be investigated; *m* stands for the number of rows of pixels of the images and *i* stands for the index of that row; *n* stands for the number of columns of pixels of the image and *j* stands for the index of that column [2].

For the processing of digital images to see if there are any errors, the mean square error is used. For determining the accuracy of an image, the two MSEs are measured and afterwards compared. In addition, the values are closer to zero are the better ones and are always non-negative.

Peak-Signal-to-Noise Ratio (PSNR) describes the ratio that falls between the power of a signal and the maximum possible value of the distortion noise which may affect the representation quality of the image. Due to the significant wideness of the image dynamic range, logarithmic decibel scale is used to express the PSNR.

PSNR's mathematical equation is as follows:

$$\mathrm{PSNR}= 20\log_{10}\left(\frac{\mathrm{MAX_f}}{\sqrt{\mathrm{MSE}}}\right) \tag{2.6}$$

Where f stands for the original image's matrix data and MAX_f signifies the quality present in the "supposed to be good" image.

Applying a filter on a test image changes the pixels of the image. The measurement of change of the pixels can be used for comparison in a systematic manner to see whether or not a certain algorithm is more effective than other ones. The peak-signal-to-noise-ratio is the metric system which is to be investigated. If there is a sort of algorithm that can degrade the image to be investigated to resemble the original image, in this case the algorithm used is determined to be the better one [2].

For the simplicity of implementation, images are considered as a *2D* array of data, or a matrix in describing the algorithms. To compare two images, their matrix dimensions should be identical.

If PSNR of a filter is high, that indicates more pixels are corrected in the filtered image. Similarly, if MSE between the filtered and non-filtered pictures are high, it shows the filter changed more pixels, and the intensity levels of the pixels are corrected more heavily.

2.11 Summary

One of the most common biometric techniques is face detection. Over recent years, several analysts have proposed distinctive face detection strategies, demanding the recognition of human faces, motivated by the expanded number of real life applications. This thesis compares the effectiveness of mean and median filters against the temporal mean, and temporal median filters, which are proposed by Dr. Bodur. So as to evaluate the effectiveness of filters on test image and the image in a dataset, mean square error (MSE) and pick signal (PSNR) were utilized in this thesis.

Chapter 3

METHODOLOGY

3.1 Introduction

This chapter will describe the research methodology. To improve face image in a video, it will depend on two filters mean and median. Then, by apply these two filters on consecutive frames in a video, in order to obtain better quality face image. Also, assessment for image quality is a traditional need. The conventional methods for measuring quality of image prior to and after improving are MSE and PSNR. In this thesis we compared different face image with different resolutions by using their quality parameters (MSE and PSNR).

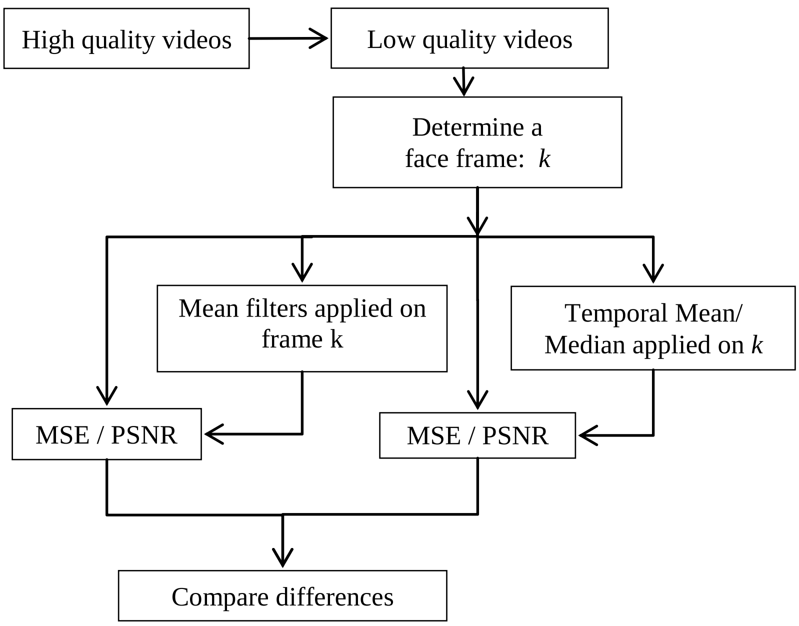

Figure 3.1: Determination of Effectiveness of Temporal Filters

3.2 Video Dataset

All of the video records used in the tests were taken in same environment with a camera at the fixed location. However, the background and the time of the day the video records were different from each other. The videos were recorded by using a camera with a resolution 1920 x 1080p. After recording the videos, they were edited, and resized into an eight different resolutions.

Table 3.1: Properties of Video Record Dataset

Total Size of Dataset	347 MB
No. of Original Video Records	50
Frame rate	29 (per/second)
Video type	.mp4
Video format	RGB24
Resolutions of Videos	1920 x 1080 480 x 640 320 x 480 288 x 352 240 x 320 120 x 240 100 x 180 80 x 150 70 x 120
Total Number of Videos	450

3.3 Determination of Face Frame *k*

Assume *V* is a low resolution video sequence and $F = \{f_1, f_2 \ldots, f_n\}$, n represents the number of frame in *V*. Firstly, the Viola-jons Algorithm has been used for detecting face by using Haar type features. Haar features are calculated at equation 2.2 as the difference between the summations of the pixels from the white region with the summation of the pixels from the black region. There are several types of Haar features. For this thesis the Haar feature that contains left eye, right eye, mouth and nose is used for detecting the full face, as described in details in the previous chapter in section 2.6. If we found the exact face, then the frame that contains a face was used for improving. Otherwise there was no face to improve. So the frame that we are selecting is our interest frame.

3.4 Mean (average) for Frame *k*

To obtain a better image of the i^{th}, f_i', a 3×3 square kernel for calculating averaging (mean) was used between pixels of frame f_i, also for the rest of frames that we selected for improving the interest frame we had to apply 3×3 square kernel. In this process 3×3, we had to calculate average for the centre pixel relies on the neighbour's pixels. We had to apply this procedure for all the pixels in the frame (*fi*), also apply it for the rest of frames (f_{i-1}, f_{i-2}, f_{i+1} , f_{i+2}) that had selected.

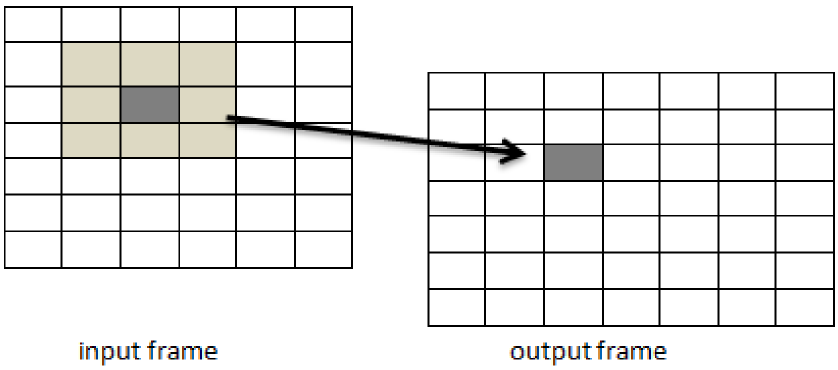

Figure 3.2: Mean filters between the pixels of frame

3.5 Applying Mean and Median Filter

After applying mean technique between the pixels of each frame, again averaging (mean) and median filters were utilizes between pixels of a sequence of framesf_{i-1}, f_{i-2}, $f_{i+1}$and f_{i+2}.

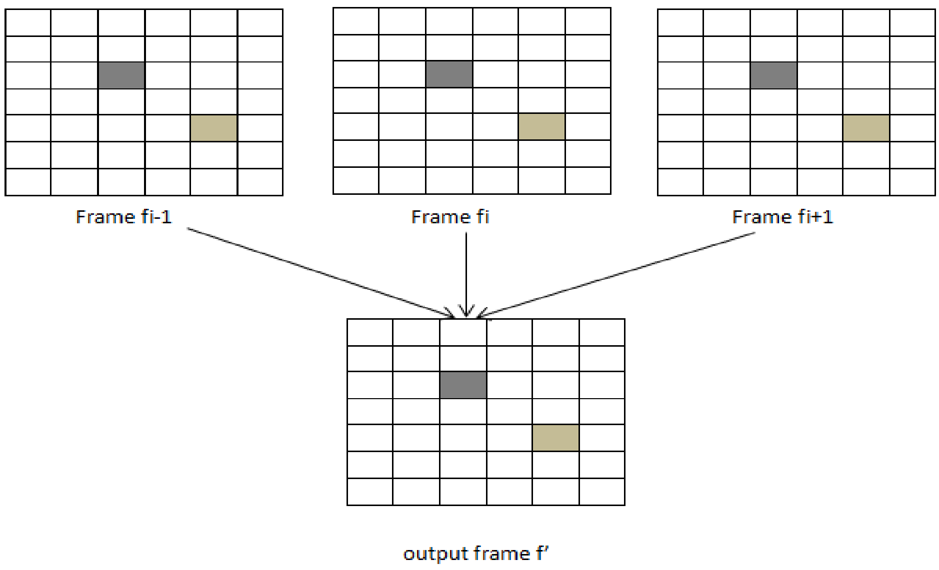

Figure 3.3: Mean / median filters between the pixels of consecutive frames

The proposed technique was calculates the mean and median for all the pixels of consequent images (frames). Therefore, all of the pixels that we are calculating with each other, from the sequence of frames, they should in the same location and putting the result to the exact position in the output image as illustrated in figure 3.3.

For this process we can easily calculate using the following formulas:

$$dn\ (i,j)= \frac{\sum_{r=m-t,..,m+t} \ d_{i,j,r}}{k} \tag{3.1}$$

$$dn\ (i,j)= \text{median}\ \{d_{i,j,m-t}, \ldots.., d_{i,j,m+t}\} \tag{3.2}$$

Where, *m* is face frame, d is the intensity value of pixel in location *i* and *j* in frame *r*, *k* is the number of frames, *t*= (k-1)/2, *dn* is the output image. The mean (average) between pixels of the different frames is calculated using (3.1). The median of consecutive neighbour pixels were calculated using (3.2).

3.6 Interpolation Method

For evaluating the difference between the images with different sizes image interpolation was used in this thesis. Image resize is a Matlab function used for changing the size of images similar to the each other; this is for measuring the difference between images before and after improving. In this thesis for measuring the performance of each low resolution with the original image we resized the original image similar to the size of low resolution image for all videos. “imresize” uses interpolation to determine the values of these pixels, computing a weighted average of some set of pixels in the vicinity of the pixel location.” imresize” bases the weightings on the distance each pixel is from the point. By default, “imresize” uses bicubic interpolation.

`W = imresize(I, scale)` returns image `W` that is scale times the size of *I*. An input image *I* can be a binary, grayscale or RGB image. If scale is from 0 through 1.0, *W* is smaller than *I*. If scale is greater than 1.0, *W* is larger than *I*. Therefore in all comparisons we reduced the size of original image similar to the size of low quality image.

3.7 Evaluating and Performance

For evaluating the rate between the low resolution image prior to and after improving with the original frame, two measures were established to determine the closeness of two frames: mean square error (MSE) and peak signal to noise ratio (PSNR). These two evaluations were described in detail in chapter two.

In this thesis the mean square error (MSE) was used for knowing the difference between the actual image and the produced image after applying our method. Also, the proposal is that the higher the PSNR, the better degraded image has been reconstructed to match the original image and the better the reconstructive algorithm. This would occur because we wish to minimize the MSE between images with respect the maximum signal value of the image.

In this thesis, there are two limitations, first one for detecting face. Correct face detection below the size 100x180 is not possible because the size is not sufficient and resizing noise makes the detection of face impossible. The effect of pixels on MSE and PSNR figures indicate that for this low resolution images the filters does not provide effective filtering.

Chapter 4

IMPLEMENTATION AND RESULTS

4.1 Implementation

In the tests, a personal computer is used with 64-bit CPU 2.10 GHz processor, 8 GB RAM, and Windows 10 operating system. The coding is implemented in Matlab 2016 because of its wide range of toolbox opportunities.

4.2 Description of Data

In this thesis 50 face videos of different persons are recorded as HD video sources. Videos are recorded by a camera with a resolution 1920 x 1080 and hold 29 frames/seconds. Each HD video is resized to generate nine different resolutions, some of them giving no face detection because resolution is extremely low. Each video is about 4 to 6 seconds. This video record database consists of brief, low resolution video clips of fifty individuals, each showing the face of a person standing in front of a camera. Each individual has an original high definition video and eight low resolution copies obtained from it by reducing the resolution as shown in Table 3.1.

As explained in previous chapters, the filter is applied on a selected frame, *k*, and its neighbour frames. At the beginning of the process all the videos are converted to sequences of frames. The image pixels of all fifty original high definition videos (1920 x 1080) of the dataset were compared to the image pixels of the rest of fifty videos in each of the low resolution dataset, they are 480 x 640, 320 x 480, 288 x

352, 240 x 320, 120 x 240, 100 x 180, 80 x 150 and 70 x 120. Then two ordinary filters, the mean and the median filters, were applied on the low resolution frames. Then, the temporal mean and median filters were accomplished all of the neighbour frames were fused by mean and median of pixel intensities. For evaluating the performance of techniques two evaluation schemes were used Mean Square Error (MSE) and Peak Signal to Noise Ratio (PSNR). After each of the mean and median operations those evaluation schemes have been used. In section 4.3 these evaluation schemes are clarified in details respectively.

4.3 Effectiveness of Filters Based on Metrics

Pixel-change effectiveness of the filters are compared using two different methods which are mostly used in scoring the quantity of pixel errors for the images after contaminated with a noise, as well as the amount of pixel corrections for an image after filtering: Mean square error (MES), and, peak signal to noise ratio (PSNR).

Especially at low resolution frames, the face images extracted from the videos needs to be improved using image filters, for an accurate biometric determination of identities. The measure of effectiveness of the filters by MSE and PSNR is assumed to provide an important indication on the quality improvement of the images.

4.4 Test Results

As described in a previous chapter, temporal filtering relies on a sequence of frames in improving the face image. We evaluate the filtered images by methods of MSE and PSNR to determine the amount of the pixel intensity changes after selecting the centre frame, *k*, and applying mean and median filters on consecutive frames.

The experiments are repeated using 3-consecutive frames, 5-consecutive frames, and 7-consecutive frames in the filtering operations. With 3-consecutive frames, after we select centre frame k, we use frames k-1 and k+1 as temporal neighbours of the frame k. Similarly, for 7-consecutive frames, we used frames $\{k\text{-}3,\ldots,k,\ldots,k+3\}$ in the temporal filter.

4.5 Temporal Filtering with 5-Consecutive Frames

The following figures and tables show the results of average for all the 50 videos with different six resolutions that depend on 2-frames before and 2-frames after the interested frame.

Tables 4.1, 4.2 and figures 4.1, 4.2 compares HD resolution frame against a) raw low resolution frames, b) low resolution frames after applying mean filter, and c) low resolution frames after applying median filter technique, this is by using two performance measurements mean square error (MSE) and peck signal to noise ratio (PSNR). Compared to the differences between the original and raw image, both temporal mean, and temporal median filtered images have less changes with respect to original. Although the percent difference of MSE is around 2%, the reduction of the difference between the original and the filtered image implies that filter has corrective effect on the image to reduce the information loss due to size reduction of the images.

Table 4.1: MSE for 5-consecutive frames around *k*

Resolutions	Original vs Raw Im.	% change	Original vs Mean	% change	Original vs Median	% change
480 x 640	6663.02	66.6%	6558.383	65.6%	6561.513	65.6%
320 x 480	4541.364	45.4%	4417.609	44.2%	4419.871	44.2%
288 x 352	7710.921	77.1%	7589.517	75.9%	7590.628	75.9%
240 x 320	6510.971	65.1%	6366.844	63.7%	6367.923	63.7%
120 x 240	103.9682	1. 04%	199.2697	2%	199.3926	2%
100 x 180	138.1957	1. 38%	240.0958	2.4%	240.377	2.4%

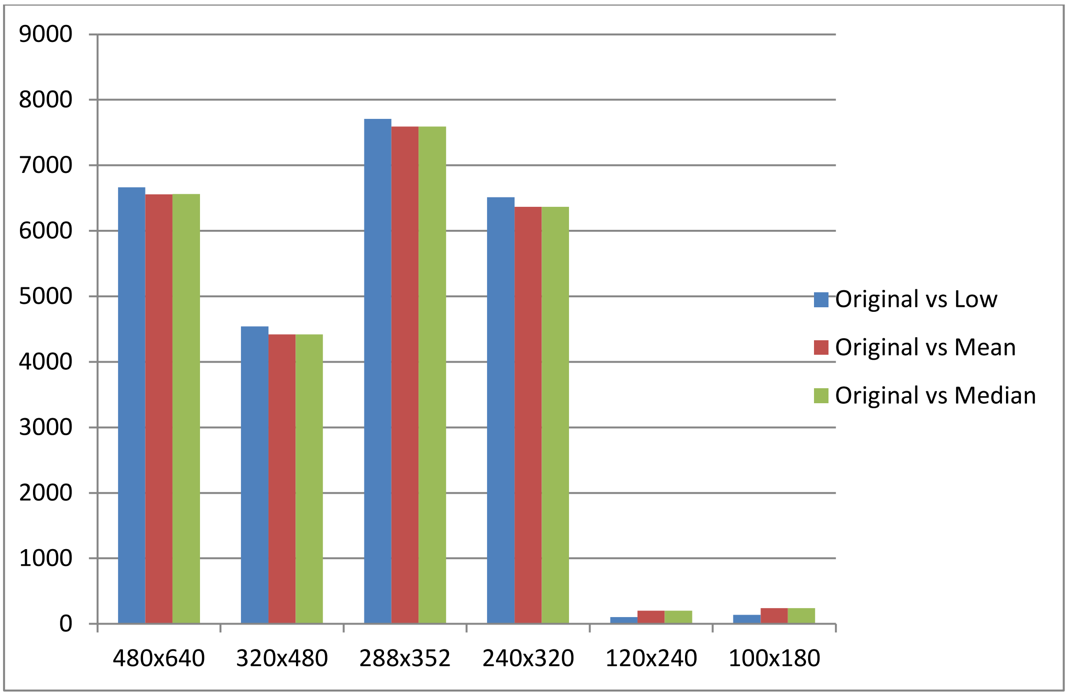

Figure 4.1: MSE for 5-Consecutive Frames around *k*

Table 4.2 :PSNR for 5-Consecutive Frames around *k*

Resolutions	Original vs Raw Image	Original vs Mean	Original vs Median
480 x 640	10.236	10.307	10.305
320 x 480	11.920	12.045	12.043
288 x 352	9.593	9.665	9.665
240 x 320	10.342	10.444	10.444
120 x 240	28.486	25.476	25.476
100 x 180	27.071	24.554	24.547

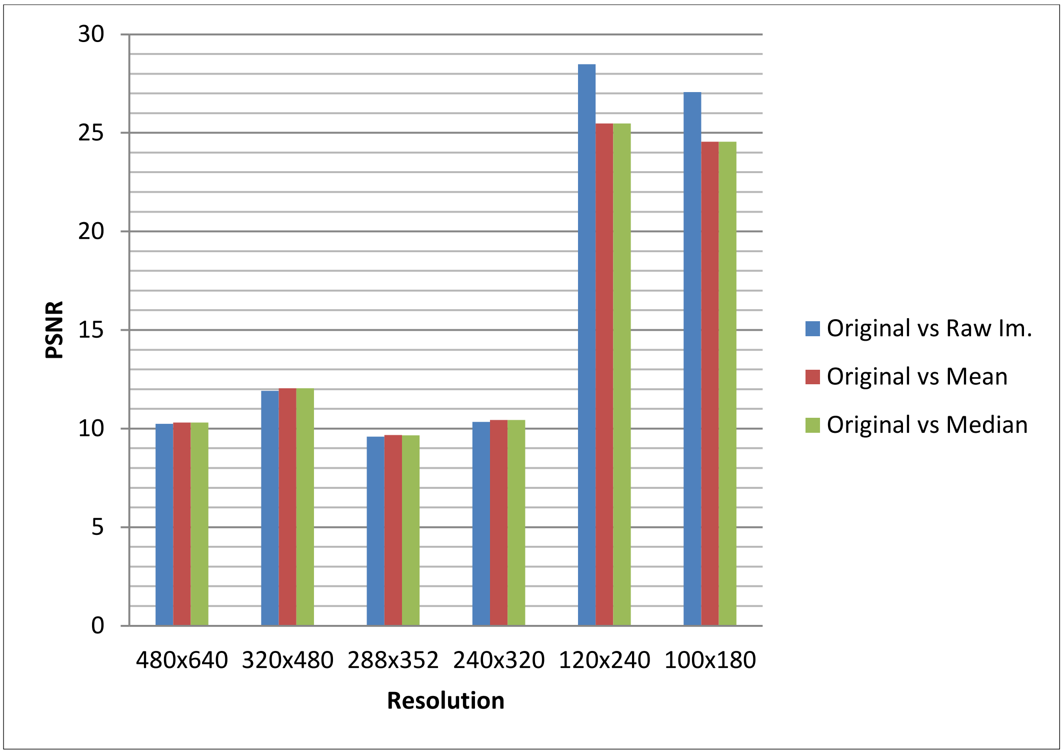

Figure 4.2: PSNR for 5 consequent frame filters

As seen by the plots of first four resolutions 480x640, 320x480, 288x352 and 240x320 improvement occurred, mean square error decreased and the peak signal-to-noise ratio increased. In contrast, in two last resolutions 120x240 and 100x180 the MSE increased and PSNR decreased probably because of extreme information loss by heavy size reduction.

Figure 4.3 shows face images that extracted from high quality video with a set of similar face images, these images were in different resolutions from higher to lower resolution, and also the images after applying mean and median techniques.

High Quality 1920 x 1080	Frame from orig Video 		
480 x 640	Frame from low video (480x640) 	Mean 	Median 
320 X 480	Frame from low video (320x480) 	Mean 	Median 
288 x 352	Frame from low video (288x352) 	Mean 	Median 
240 x 320	Frame from low video (240x320) 	Mean 	Median 

120 x 240			
100 x 180			

Figure 4.3: Remaining Frames Before and After Filtering

In general, only small differences occur between the values of mean and median filters. In case, mean filter had better improve than the median filter for 2-frames before and 2-frames after the selected frame (k).

4.6 Temporal Filtering with 7-Consecutive Frames

The test procedure for 5-consecutive frames is repeated to get the average for all the 50 videos for six different resolutions using 3 previous frames, 3 post frames starting from the centre frame *k*, thus processing 7-consecutive frames of the video records.

Tables 4.3, 4.4 and figures 4.3, 4.4 consists of the test results of comparisons for a) the difference of original frames to the raw low resolution frames, b) the difference of original frames to the filtered low resolution frames by temporal mean, c) similar to (b) but using temporal median filter. The differences are evaluated by two methods, MSE and PSNR.

Table 4.3: MSE for temporal filters with 7 consecutive frames around *k*

Resolutions	Original vs Low	by %	Original vs Mean	By %	Original vs Median	By %
480 x 640	6663.02	66.63%	6555.49	65.55%	6559.45	65.59%
320 x 480	4541.36	45.41%	4415.24	44.14%	4418.24	44.18%
288 x 352	7710.92	77.11%	7588.29	75.88%	7589.86	75.9%
240 x 320	6510.97	65.11%	6494.21	64.94%	6495.86	64.96%
120 x 240	103.968	1.4%	201.20	2.01%	201.28	2.01%
100 x 180	138.2	1.38%	240.93	2.04%	241.02	2.41%

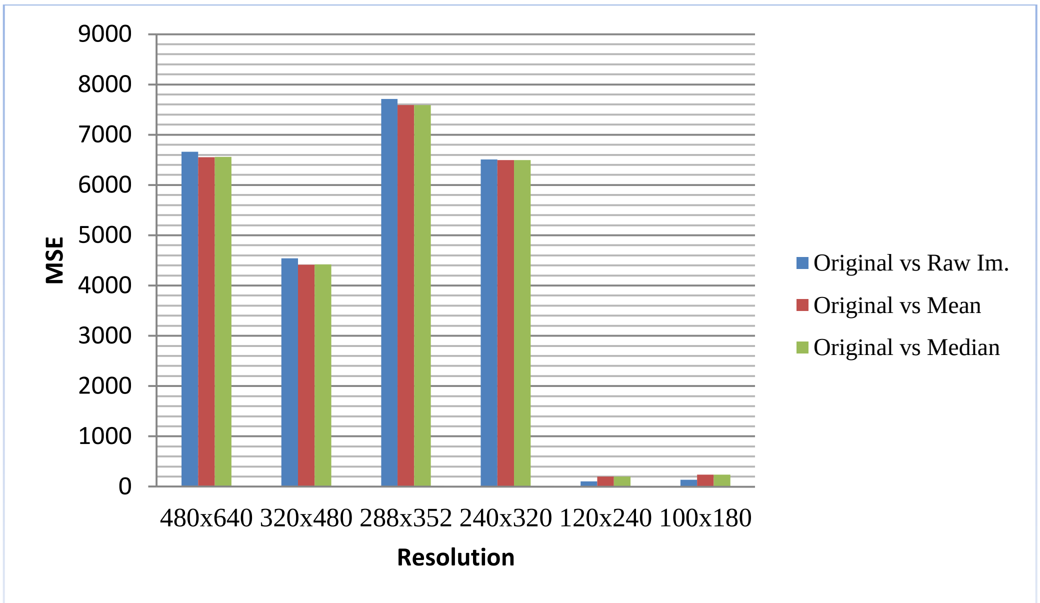

Figure 4.4: MSE for temporal filters with 7 consecutive frames around *k*

Table 4.4: PSNR for temporal filters with 7 consecutive frames around *k*

Resolutions	Original vs Low	Original vs Mean	Original vs Median
480 x 640	10.235861	10.3095	10.3066519
320 x 480	11.92019	12.04815	12.04482
288 x 352	9.592557	9.666018	9.665083
240 x 320	10.2838	10.38566	10.38449
120 x 240	28.48555	25.42783	25.42564
100 x 180	27.0707	24.53569	24.53305

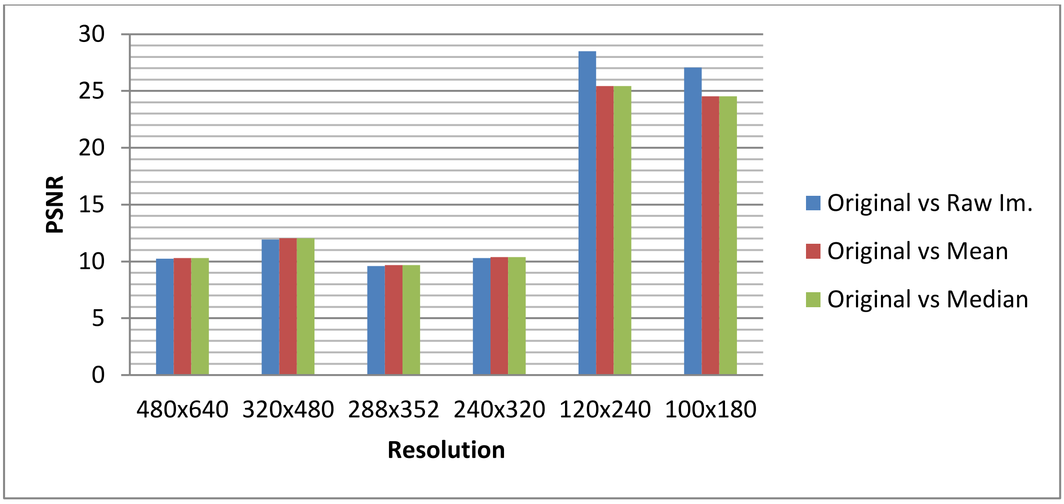

Figure 4.5: PSNR for temporal filters with 7 consecutive frames around *k*

As seen on the graph, bars for 5 consequent frame filters of first four qualities 480x640, 320x480, 288x352 and 240x320 has almost same the bar of raw filter PSNR, mean square error decreased and the peak signal-to-noise-ratio increased, in contrast, in last two resolutions 120x240 and 100x180 the MSE increased and PSNR decreased.

Generally, small differences occur for the values of MSE between mean and median filters, also same differences in PSNR values between the two techniques. In general, mean filter had more effect than the median filter for 5 consecutive frame temporal filters.

The results show that for the first three resolutions 7 consecutive frames had more effect compared to 5 consecutive frame filters. On the other hand, for the last improved resolution (240 x 320), 5 consecutive frame filtering had higher effect than the other filter. This is illustrates, for the very low resolution frames, fusing the less number of frames will obtain better improvement.

In this thesis, temporal mean and median filters with 5 and 7 consecutive frames were applied on the pixels of each selected frame separately, and then mean (average) and median filters were applied between the pixels of sequence frames. The results illustrate that positive effect of the applied mean and median filters were observed for the images with higher first four resolutions. Although the difference between the temporal mean and median filters are not significant, mean filter had slightly higher effect on the tested face images.

Chapter 5

CONCLUSION

Improving of face image is necessary to use low resolution image in face detection and recognition. This thesis tested the effects of two temporal image improving techniques for filtering face images in a large range of resolutions.

The temporal filters are a good candidate for improving images for the face images even near the lower bound of resolution. The temporal mean and the median filters are an expansion of ordinary mean and median filters to temporal domain to be used on consecutive images captured from a video record. In the tests for fifty faces, they both decreased the error rate of biometrics obtained from low resolution videos using the consecutive images around a certain image frame.

The thesis indicates that intensity level change of pixels of the non filtered low resolution images with respect to original image is more than the temporal filtered low resolution images with respect to original image. This shows that the temporal filtering recovers some information from the consequent frames, and improves the quality of the face image. Simplicity of the filter, and its computationally inexpensive algorithm can be listed as some of the important advantages of applying temporal mean and temporal median filtering on a sequence of frames.

Tests indicate that the quality of the output image through temporal mean filtering on a consecutive set of frames is better than that of the temporal median filtering technique.

Future studies may be recommended on testing the success rate improvement of face detection using temporal filters against using the raw and ordinary filters. The MSE and PSNR values in the tests indicate that an improvement of successful face detection rate is expectable.

REFERENCES

[1] Huang, T., & Wang, Z. Face Detection Using Improved AdaBoost. *New York University, USA.*

[2] Measures of image quality (2016, December, 10). Retrieved From. http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/VELDHUIZEN/node18.html.

[3] Jillela, R. R., Ross, A., Li, X., & Adjeroh, D. (2008). *Adaptive Frame Selection for Enhanced Face Recognition in Low-Resolution Videos*. West Virginia University Libraries.

[4] Alionte, E., & Lazar, C. (2015, October). A practical implementation of face detection by using Matlab cascade object detector. In *System Theory, Control and Computing (ICSTCC), 2015 19th International Conference on* (pp. 785-790). IEEE.

[5] Yang, M. H., Kriegman, D. J., & Ahuja, N. (2002). Detecting faces in images: A survey. *IEEE Transactions on pattern analysis and machine intelligence*, *24*(1), 34-58.

[6] Lehmann, T. M., Gonner, C., & Spitzer, K. (1999). Survey: Interpolation methods in medical image processing. *IEEE transactions on medical imaging*, *18*(11), 1049-1075.

[7] Hassaballah, M., & Aly, S. (2015). Face recognition: challenges, achievements and future directions. *IET Computer Vision*, *9*(4), 614-626.

[8] Baker, B. D., & Gunter, W. D. (2005). Surveillance: concepts and practices for fraud, security and crime investigation. *Int. Found. Prot. Off*, *2*, 1-17.

[9] Types of Survelillance: Camera, Telephones etc. (2016, Nov, 3). Retrieved From. http://www.wsystems.com/news/surveillance-cameras-types.html

[10]Biometric surveillance: searching for identity. (2016, Nov. 1). Retrieved From. https://business.highbeam.com/127/article-1G1-81471034/biometric-surveillance-searching-identity

[11]Ovsenik, L., Kolesárová, A. K., & Turán, J. (2010). Video surveillance systems. *Acta Electrotechnica et Informatica*, *10*(4), 46-53.

[12]Cisco Video Surveillance operations Manager. (2014, Jan, 17). Retrieved From. http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Education/SchoolsSRA_DG/SchoolsSRA-DG/SchoolsSRA_chap8.html.

[13]Jillela, R. R., & Ross, A. (2009, June). Adaptive frame selection for improved face recognition in low-resolution videos. In 2009 International Joint Conference on Neural Networks (pp. 1439-1445). IEEE.

[14]Face Recognition: An Introducion. (2016, October, 5). Retrieved From. https://alitarhini.wordpress.com/tag/face-recognition/.

[15]GTAV Face Database. (2016, October, 20). Retrieved From. https://francesctarres.wordpress.com/gtav-face-database/.

[16]Expressions (2016, October, 19). Retrieved From. http://www.quizz.biz/quizz-430184.html.

[17]Naseem, I., Togneri, R., & Bennamoun, M. (2010). Linear regression for face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(11), 2106-2112.

[18]Islam, M. M., Asari, V. K., Islam, M. N., & Karim, M. A. (2010). Super-resolution enhancement technique for low resolution video. *IEEE Transactions on Consumer Electronics*, *56*(2), 919-924.

[19]Sundaram, D. K. M., Sasikala, D., & Rani, P. A. (2014). A study on preprocessing a mammogram image using Adaptive Median Filter. *International Journal of Innovative Research in Science, Engineering and Technology*, *3*(3), 10333-10337.

[20]Khadhraoui, T., Benzarti, F., & Amiri, H. Robust Facial Feature Detection for Registration.

[21]Biometric. (2016, December, 15). Retrieved From. http://www.globalsecurity.org/security/systems/biometrics.htm.

[22]Chao, W. L. (2007). Face Recognition. *GICE, National Taiwan University*.

APPENDICES

Appendix A: Results of 2-Frames Before and 2-After Frame k

No. of video	MSE Original VS 480*640	MSE Original VS 480*640 mean Filter	MSE Original VS 480*640 median Filter	PSNR Original VS 480*640	PSNR Original VS 480*640 mean Filter	PSNR Original VS 480*640 median Filter
1	7023.435	6917.709	6921.103	9.665308	9.73118	9.72905
2	3975.383	3925.681	3927.977	12.13701	12.19165	12.18911
3	3810.412	3734.672	3746.622	12.32108	12.40828	12.39441
4	6984.566	6916.712	6919.669	9.689409	9.731807	9.72995
5	6215.686	6119.868	6125.097	10.19591	10.26338	10.25967
6	7510.514	7396.148	7406.467	9.374107	9.440748	9.434693
7	7053.742	6916.719	6929.263	9.646608	9.731802	9.723933
8	8155.101	7979.403	7980.923	9.01651	9.1111	9.110272
9	6290.807	6242.401	6243.321	10.14374	10.17729	10.17665
10	2916.644	2878.106	2881.83	13.48197	13.53974	13.53412
11	6482.258	6423.425	6425.61	10.01354	10.05314	10.05166
12	2472.176	2439.867	2441.054	14.20001	14.25714	14.25503
13	9398.735	9310.277	9315.914	8.400109	8.441178	8.438549
14	2336.322	2294.244	2295.344	14.44548	14.52441	14.52233
15	6620.621	6542.308	6544.757	9.921817	9.973494	9.971868
16	1184.733	1157.583	1158.635	17.3946	17.49528	17.49134
17	9625.02	9538.495	9538.805	8.296787	8.336005	8.335864
18	10119.99	9769.348	9772.19	8.079002	8.232148	8.230884
19	9350.03	9239.255	9239.565	8.422674	8.474434	8.474288
20	6177.567	6077.729	6078.723	10.22263	10.29339	10.29268
21	10287.72	10204.28	10206.69	8.007614	8.042981	8.041955
22	11092.26	11005.55	11007.56	7.680605	7.714685	7.713892
23	11218.23	11097.86	11102.55	7.631561	7.678413	7.676578
24	10755.84	10661.53	10663.77	7.814362	7.852607	7.851697
25	6614.862	6509.985	6510.344	9.925596	9.995004	9.994764
26	7685.289	7298.761	7313.63	9.274201	9.498312	9.489474
27	5413.015	5233.662	5234.246	10.79641	10.94275	10.94226
28	6314.218	6203.826	6202.149	10.12761	10.20421	10.20538
29	9678.944	9575.384	9576.652	8.272524	8.319242	8.318667
30	3672.576	3569.025	3570.91	12.4811	12.60531	12.60301
31	5814.744	5754.082	5757.418	10.4855	10.53104	10.52853
32	6421.594	6371.066	6372.188	10.05438	10.08868	10.08792
33	6390.468	6114.283	6117.55	10.07548	10.26735	10.26503
34	8852.171	8753.458	8756.673	8.660306	8.709007	8.707412
35	5110.768	5084.259	5086.309	11.04594	11.06853	11.06678
36	7125.451	6953.347	6964.293	9.60268	9.708864	9.702034
37	5602.914	5536.17	5538.614	10.64666	10.69871	10.69679
38	9751.748	9633.738	9633.375	8.239979	8.292855	8.293019
39	3487.18	3411.428	3413.129	12.70606	12.80144	12.79928
40	7755.767	7693.632	7695.999	9.234556	9.26949	9.268154
41	6965.886	6869.269	6875.708	9.70104	9.761698	9.757629
42	4365.726	4228.021	4239.685	11.73024	11.86943	11.85747
43	6232.423	6083.696	6086.914	10.18423	10.28913	10.28683
44	4419.999	4369.374	4369.943	11.67658	11.72661	11.72605
45	5666.324	5567.315	5571.796	10.59779	10.67435	10.67085
46	7271.799	7112.592	7115.335	9.514385	9.610525	9.60885
47	5776.753	5721.515	5719.122	10.51397	10.55569	10.55751
48	5186.04	5146.882	5148.399	10.98244	11.01536	11.01408
49	7698.83	7606.35	7601.486	9.266556	9.31904	9.321818
50	6817.749	6728.856	6730.361	9.794394	9.851391	9.85042

No. of video	MSE Original VS 320*480	MSE Original VS 320*480 mean Filter	MSE Original VS 320*480 median Filter	PSNR Original VS 320*480	PSNR Original VS 320*480 mean Filter	PSNR Original VS 320*480 median Filter
1	4515.144	4392.657	4397.665	11.58409	11.70353	11.69858
2	2852.288	2787.941	2788.868	13.57887	13.67797	13.67652
3	2565.699	2479.395	2489.763	14.03875	14.18735	14.16922
4	4589.271	4499.356	4500.215	11.51337	11.5993	11.59847
5	4037.522	3914.209	3918.818	12.06965	12.20436	12.19925
6	5102.599	4958.5	4965.04	11.05289	11.1773	11.17158
7	4742.565	4579.406	4588.48	11.37067	11.52271	11.51412
8	5792.73	5574.317	5573.678	10.50197	10.66889	10.66938
9	4323.011	4254.682	4254.805	11.77294	11.84213	11.84201
10	1772.325	1725.996	1729.028	15.64537	15.76041	15.75278
11	4152.98	4076.031	4078.576	11.94721	12.02843	12.02572
12	1595.359	1552.19	1553.871	16.10222	16.22136	16.21665
13	6441.485	6321.24	6324.907	10.04094	10.12278	10.12026
14	1550.104	1504.07	1505.003	16.2272	16.35812	16.35543
15	4691.428	4582.084	4584.038	11.41775	11.52017	11.51832
16	747.2317	714.9314	715.3566	19.39625	19.58816	19.58558
17	6728.97	6603.351	6603.636	9.851318	9.93316	9.932972
18	7104.386	6828.718	6831.231	9.615538	9.787412	9.785814
19	6528.531	6373.609	6374.568	9.982649	10.08695	10.0863
20	4099.319	3978.175	3978.711	12.00369	12.13396	12.13338
21	6886.926	6770.598	6773.628	9.750549	9.824533	9.82259
22	7075.439	6964.996	6965.423	9.63327	9.701595	9.701328
23	7567.645	7431.513	7436.017	9.341196	9.420031	9.4174
24	7215.038	7083.977	7086.174	9.548417	9.628032	9.626686
25	4521.411	4408.389	4408.554	11.57806	11.688	11.68784
26	5231.601	4900.172	4909.193	10.94446	11.22869	11.2207
27	3956.953	3737.785	3738.697	12.15719	12.40466	12.4036
28	4384.482	4240.869	4239.441	11.71162	11.85626	11.85772
29	6470.402	6333.039	6335.059	10.02149	10.11468	10.1133
30	2407.348	2284.303	2283.443	14.31542	14.54327	14.5449
31	3847.039	3768.854	3772.317	12.27954	12.36871	12.36472
32	4191.242	4130.398	4131.6	11.90738	11.97088	11.96962
33	4414.639	4075.348	4076.936	11.68185	12.02916	12.02746
34	6003.563	5876.943	5879.837	10.34671	10.43929	10.43715
35	3266.606	3232.13	3234.323	12.98984	13.03591	13.03297
36	4909.242	4721.619	4728.471	11.22066	11.38989	11.3836
37	3890.032	3812.682	3813.08	12.23127	12.3185	12.31804
38	6726.935	6579.856	6579.491	9.852631	9.94864	9.948881
39	2382.149	2280.955	2283.317	14.36111	14.54964	14.54514
40	5177.124	5094.502	5097.444	10.98992	11.05979	11.05728
41	4842.627	4717.824	4722.458	11.27999	11.39339	11.38912
42	2943.078	2806.06	2812.705	13.44279	13.64983	13.63956
43	4451.675	4281.679	4283.769	11.64557	11.81466	11.81254
44	2861.052	2804.226	2804.789	13.56555	13.65267	13.6518
45	4227.671	4107.84	4108.089	11.86979	11.99467	11.99441
46	5728.694	5518.26	5520.616	10.55025	10.71278	10.71093
47	3777.721	3711.223	3709.379	12.3585	12.43563	12.43779
48	3572.161	3528.877	3530.268	12.60149	12.65444	12.65273
49	5388.768	5276.247	5274.382	10.81591	10.90755	10.90909
50	4815.997	4698.449	4698.395	11.30394	11.41126	11.41131

No. of video	MSE Original VS 288*352	MSE Original VS 288*352 mean Filter	MSE Original VS 288*352 median Filter	PSNR Original VS 288*352	PSNR Original VS 288*352 mean Filter	PSNR Original VS 288*352 median Filter
1	8329.208	8170.987	8170.679	8.924766	9.008058	9.008222
2	4626.298	4561.696	4562.176	11.47847	11.53954	11.53908
3	4468.057	4400.16	4405.839	11.62962	11.69612	11.69052
4	8286.931	8186.302	8185.814	8.946866	8.999926	9.000185
5	7522.795	7400.383	7398.415	9.367011	9.438262	9.439417
6	8627.618	8478.226	8483.316	8.771894	8.847754	8.845147
7	8207.745	8062.746	8067.15	8.988565	9.065974	9.063602
8	9253.037	9037.928	9037.718	8.46796	8.570115	8.570216
9	7420.335	7347.268	7348.244	9.426569	9.469545	9.468968
10	3543.019	3491.817	3493.102	12.63707	12.70029	12.69869
11	7637.179	7553.342	7553.781	9.301474	9.349412	9.349159
12	2952.308	2909.128	2909.539	13.42919	13.49318	13.49256
13	10909.67	10776.92	10783.27	7.752688	7.805856	7.803298
14	2787.723	2744.837	2744.349	13.67831	13.74564	13.74641
15	7675.131	7560.8	7563.416	9.279945	9.345126	9.343624
16	1392.018	1355.759	1354.926	16.69436	16.80898	16.81165
17	10670.08	10543.41	10545.34	7.849125	7.900994	7.900196
18	11123.51	10945.17	10947.21	7.668385	7.738578	7.73777
19	10720.95	10570.73	10571.85	7.828471	7.889752	7.889295
20	7374.524	7242.124	7242.906	9.453464	9.532144	9.531675
21	12013.1	11884.75	11885.35	7.334252	7.380904	7.380682
22	12984.3	12861.95	12863.97	6.996619	7.037735	7.037055
23	12903	12765.18	12765.6	7.023896	7.070535	7.070391
24	12533.17	12397.34	12400.22	7.150193	7.19752	7.196509
25	7998.666	7892.701	7890.883	9.100628	9.158547	9.159547
26	8443.88	8218.844	8222.349	8.865383	8.982696	8.980845
27	5997.386	5793.704	5794.152	10.35118	10.50124	10.5009
28	7333.661	7177.517	7179.293	9.477596	9.571061	9.569987
29	11262.83	11121.32	11122.35	7.614328	7.669239	7.668837
30	4306.249	4180.147	4179.878	11.78981	11.91889	11.91917
31	6874.998	6796.23	6800.293	9.758078	9.808123	9.805528
32	7850.089	7777.62	7777.726	9.182058	9.222336	9.222277
33	7268.558	7038.41	7039.366	9.516321	9.656058	9.655468
34	10325.1	10193.38	10194.54	7.991859	8.04762	8.047127
35	6161.737	6120.222	6120.708	10.23377	10.26313	10.26279
36	8197.01	8013.979	8016.538	8.994249	9.092322	9.090935
37	6428.286	6344.138	6344.156	10.04985	10.10708	10.10707
38	11326.88	11167.36	11166.43	7.589699	7.651297	7.65166
39	3953.946	3845.972	3845.589	12.1605	12.28074	12.28118
40	9276.902	9190.558	9190.055	8.456774	8.497385	8.497622
41	7858.219	7750.586	7756.213	9.177562	9.237458	9.234306
42	4991.504	4852.382	4854.894	11.14849	11.27125	11.26901
43	6856.455	6677.679	6678.556	9.769807	9.884548	9.883978
44	5173.071	5112.303	5112.4	10.99332	11.04464	11.04456
45	6114.044	5987.783	5988.154	10.26752	10.35814	10.35787
46	8041.628	7805.192	7805.344	9.077364	9.206968	9.206883
47	6843.203	6763.546	6761.686	9.778209	9.829059	9.830254
48	5970.969	5920.371	5920.999	10.37036	10.40731	10.40685
49	8948.566	8833.854	8832.116	8.613269	8.669301	8.670156
50	7780.501	7651.085	7652.569	9.220728	9.293574	9.292731

No. of video	MSE Original VS 240*320	MSE Original VS 240*320 mean Filter	MSE Original VS 240*320 median Filter	PSNR Original VS 240*320	PSNR Original VS 240*320 mean Filter	PSNR Original VS 240*320 median Filter
1	6818.437	6642.006	6642.409	9.793955	9.907811	9.907547
2	3916.266	3833.081	3833.667	12.20208	12.29532	12.29466
3	3734.524	3641.996	3647.381	12.40845	12.51741	12.51099
4	6927.967	6807.632	6807.429	9.724745	9.800843	9.800973
5	6096.723	5954.425	5953.231	10.27984	10.38241	10.38328
6	7373.943	7199.559	7204.784	9.453806	9.557745	9.554594
7	6860.671	6700.081	6706.279	9.767138	9.870003	9.865988
8	7917.428	7657.368	7659.903	9.144962	9.290009	9.288571
9	6241.192	6148.328	6149.153	10.17813	10.24323	10.24265
10	2859.524	2800.849	2801.943	13.56787	13.65791	13.65621
11	6411.285	6305.63	6305.542	10.06135	10.13352	10.13358
12	2402.557	2349.216	2349.979	14.32407	14.42157	14.42016
13	9301.332	9140.836	9145.367	8.445352	8.520945	8.518792
14	2265.262	2221.386	2221.344	14.57962	14.66456	14.66465
15	6556.874	6418.024	6419.214	9.963836	10.05679	10.05598
16	1145.332	1103.679	1102.736	17.54149	17.70238	17.70609
17	9523.253	9353.442	9353.616	8.342951	8.421089	8.421008
18	9525.107	9304.565	9305.904	8.342105	8.443843	8.443218
19	9248.697	9057.756	9057.913	8.469998	8.560597	8.560522
20	6085.222	5934.393	5934.231	10.28804	10.39704	10.39716
21	10188.09	10027.75	10028.26	8.049877	8.118768	8.118546
22	10970.79	10820.63	10823.08	7.728424	7.788277	7.787296
23	11034.95	10873.09	10876.08	7.703102	7.767274	7.766078
24	10645.23	10476.34	10478.61	7.859253	7.928709	7.927767
25	6401.2	6271.638	6270.696	10.06819	10.15699	10.15765
26	7013.365	6795.846	6799.824	9.671539	9.808368	9.805827
27	5145.25	4922.806	4922.841	11.01674	11.20868	11.20865
28	6179.726	6003.964	6005.318	10.22111	10.34642	10.34544
29	9508.663	9330.362	9332.261	8.349609	8.431818	8.430935
30	3528.757	3394.05	3392.779	12.65459	12.82362	12.82525
31	5719.999	5620.946	5622.705	10.55684	10.63271	10.63135
32	6351.835	6265.393	6265.76	10.10181	10.16132	10.16107
33	6035.879	5806.895	5807.727	10.3234	10.49136	10.49074
34	8765.07	8599.734	8600.765	8.70325	8.785953	8.785433
35	5017.623	4959.906	4960.333	11.12582	11.17607	11.1757
36	6854.841	6644.083	6648.053	9.77083	9.906453	9.903859
37	5420.544	5314.501	5313.557	10.79037	10.87618	10.87695
38	9578.438	9384.749	9386.075	8.317857	8.406577	8.405963
39	3387.271	3256.954	3255.835	12.8323	13.00269	13.00418
40	7673.105	7559.824	7561.929	9.281092	9.345687	9.344478
41	6862.212	6719.548	6720.494	9.766163	9.857403	9.856791
42	4117.621	3968.25	3969.448	11.98434	12.14481	12.1435
43	5982.123	5787.175	5788.641	10.36225	10.50614	10.50504
44	4315.366	4237.418	4236.933	11.78063	11.85979	11.86029
45	5526.394	5365.138	5365.15	10.70639	10.83499	10.83499
46	7091.484	6823.274	6823.598	9.623432	9.790876	9.790669
47	5667.773	5566.585	5565.308	10.59668	10.67491	10.67591
48	5074.007	5004.671	5005.201	11.07729	11.13705	11.13659
49	7622.152	7470.684	7470.593	9.310028	9.3972	9.397253
50	6657.199	6495.722	6496.259	9.897888	10.00453	10.00417

No. of video	MSE Original VS 120*240	MSE Original VS 120*240 mean Filter	MSE Original VS 120*240 median Filter	PSNR Original VS 120*240	PSNR Original VS 120*240 mean Filter	PSNR Original VS 120*240 median Filter
1	110.3409	190.7996	190.4287	27.70344	25.32503	25.33348
2	50.92693	104.5933	104.5766	31.06133	27.93576	27.93646
3	54.90789	127.7123	123.047	30.73446	27.06848	27.23009
4	54.20314	131.6684	132.4316	30.79056	26.93599	26.91089
5	123.6192	219.5291	219.7427	27.20995	24.71588	24.71166
6	81.87874	193.5309	193.7507	28.99909	25.2633	25.25837
7	121.3944	215.0902	216.9489	27.28882	24.8046	24.76723
8	120.6526	279.3742	279.9215	27.31544	23.66894	23.66044
9	40.4589	110.1318	110.2137	32.06066	27.71167	27.70845
10	61.40105	95.69013	95.69198	30.24905	28.32213	28.32205
11	69.63684	131.701	131.4643	29.70241	26.93491	26.94273
12	105.2392	142.8067	142.4537	27.90903	26.58332	26.59406
13	55.66336	165.0047	164.5029	30.67511	25.95584	25.96907
14	58.92306	85.80472	85.96101	30.42795	28.79569	28.78779
15	62.67817	145.6037	145.5634	30.15964	26.49908	26.50028
16	54.38493	73.93294	73.8457	30.77602	29.44242	29.44755
17	60.9612	166.8836	166.8781	30.28027	25.90667	25.90681
18	228.4366	355.7537	355.5483	24.54315	22.61931	22.62182
19	78.7151	187.6314	188.0007	29.17022	25.39775	25.38921
20	175.6349	249.0423	249.1725	25.68469	24.16807	24.1658
21	84.18807	199.1498	198.5354	28.8783	25.139	25.15242
22	63.19544	166.3121	166.3735	30.12395	25.92157	25.91996
23	67.18953	178.8404	178.7526	29.85779	25.60615	25.60828
24	142.682	252.0873	252.3828	26.58711	24.11529	24.11021
25	147.9987	247.2492	246.825	26.42823	24.19945	24.20691
26	318.0552	489.8382	495.0252	23.10578	21.23028	21.18453
27	135.8505	286.3431	286.3669	26.80019	23.56194	23.56157
28	101.858	204.8172	204.8718	28.05085	25.01714	25.01598
29	73.28317	192.9617	192.9438	29.48076	25.27609	25.2765
30	98.32815	152.2795	152.7724	28.20403	26.30439	26.29036
31	58.72855	132.4016	132.2138	30.44231	26.91187	26.91804
32	62.77095	137.3378	137.5209	30.15322	26.7529	26.74712
33	291.6337	441.7182	442.208	23.48243	21.67935	21.67454
34	62.35424	167.6949	167.5182	30.18214	25.8856	25.89018
35	65.71288	125.1083	125.351	29.9543	27.15794	27.14953
36	181.2225	326.4336	328.0766	25.54868	22.99286	22.97105
37	124.2567	213.9519	213.4182	27.1876	24.82764	24.83849
38	78.68741	223.5373	223.5846	29.17175	24.6373	24.63638
39	100.5336	195.6347	195.5531	28.10769	25.21635	25.21816
40	59.66878	151.7028	151.7071	30.37333	26.32087	26.32074
41	110.9965	197.7368	198.1534	27.67771	25.16993	25.16079
42	124.0266	232.5749	231.9014	27.19565	24.46517	24.47777
43	100.5894	192.8235	192.2882	28.10528	25.2792	25.29128
44	90.78376	136.9231	136.965	28.55072	26.76604	26.76471
45	138.3309	224.859	224.7548	26.72161	24.6117	24.61371
46	197.6829	398.68	398.7705	25.17111	22.12456	22.12357
47	91.93579	173.2187	173.1147	28.49596	25.74486	25.74746
48	82.98608	148.0156	147.6673	28.94075	26.42773	26.43796
49	46.75583	157.8231	161.1141	31.43245	26.1491	26.05947
50	126.0693	243.146	242.7542	27.12471	24.27213	24.27914

No. of video	MSE Original VS 100*180	MSE Original VS 100*180 mean Filter	MSE Original VS 100*180 median Filter	PSNR Original VS 100*180	PSNR Original VS 100*180 mean Filter	PSNR Original VS 100*180 median Filter
1	169.5264	267.332	267.3942	25.83843	23.86029	23.85928
2	72.48956	135.7239	135.6718	29.52805	26.80424	26.80591
3	73.38513	153.2847	157.6964	29.47472	26.27582	26.15258
4	77.62631	166.7135	166.9623	29.23071	25.9111	25.90462
5	185.2211	278.6952	278.703	25.4539	23.67951	23.67939
6	128.2671	244.2173	243.6925	27.04965	24.25304	24.26238
7	165.7835	262.8873	264.252	25.93539	23.93311	23.91062
8	175.0817	342.2518	342.3649	25.6984	22.78735	22.78591
9	57.34861	143.8285	143.6437	30.54557	26.55236	26.55794
10	88.28781	126.9019	127.1779	28.6718	27.09612	27.08669
11	101.0426	168.3989	168.7324	28.08576	25.86741	25.85882
12	132.291	172.9094	173.1609	26.9155	25.75262	25.74631
13	83.41328	212.2644	211.753	28.91845	24.86203	24.87251
14	72.84743	102.7733	102.6634	29.50666	28.012	28.01665
15	93.34913	185.2492	185.0643	28.4297	25.45324	25.45758
16	83.02619	102.1585	102.1808	28.93865	28.03806	28.03711
17	96.09224	218.4435	218.4956	28.30392	24.73741	24.73638
18	169.4533	311.362	311.3682	25.8403	23.19815	23.19806
19	116.1016	238.852	238.9453	27.48242	24.34952	24.34782
20	224.2645	307.5992	308.8081	24.6232	23.25095	23.23392
21	126.5063	255.6856	255.9945	27.10968	24.05374	24.0485
22	102.5178	225.8156	226.6609	28.02281	24.59326	24.57704
23	92.24726	219.9852	220.3039	28.48127	24.70687	24.70058
24	196.8314	319.3788	319.5032	25.18986	23.08774	23.08605
25	184.8292	295.297	294.9846	25.4631	23.42821	23.43281
26	311.2747	412.1395	411.9821	23.19936	21.98036	21.98202
27	166.3901	300.413	300.3327	25.91953	23.35362	23.35478
28	152.1501	264.381	264.35	26.30808	23.9085	23.90901
29	120.4875	253.0508	253.0763	27.32138	24.09873	24.09829
30	152.4803	204.1365	204.2177	26.29867	25.0316	25.02987
31	88.12681	171.0812	170.9138	28.67972	25.79878	25.80303
32	89.36478	172.605	173.0158	28.61914	25.76027	25.74994
33	283.5354	358.749	358.8625	23.60473	22.5829	22.58152
34	92.70709	217.6018	217.643	28.45967	24.75418	24.75336
35	96.54204	164.3309	165.7723	28.28364	25.97361	25.93568
36	240.0366	389.3853	390.039	24.32803	22.22701	22.21972
37	158.3439	264.6765	264.855	26.13479	23.90365	23.90072
38	119.4391	277.8071	277.6731	27.35934	23.69337	23.69547
39	146.0179	248.2702	248.438	26.48674	24.18156	24.17862
40	91.269	199.1005	199.7746	28.52757	25.14008	25.1254
41	145.2585	244.5836	244.8928	26.50939	24.24653	24.24104
42	180.9114	293.5978	292.9156	25.55615	23.45328	23.46338
43	153.5281	249.9095	249.1605	26.26893	24.15298	24.16601
44	128.1993	182.054	182.0445	27.05195	25.5288	25.52903
45	182.8213	279.989	279.979	25.51054	23.65939	23.65955
46	265.7172	478.753	478.4235	23.88661	21.32969	21.33268
47	131.4914	222.1935	221.7999	26.94183	24.66349	24.67119
48	104.7841	183.9062	183.8531	27.92785	25.48484	25.48609
49	66.49611	206.0045	210.0261	29.90284	24.99204	24.90807
50	174.5821	308.0604	308.6294	25.71081	23.24444	23.23643

Appendix B: Results of 3-Frames Before and 3-After Frame k

No. of video	MSE Original VS 480*640	MSE Original VS 480*640 mean Filter	MSE Original VS 480*640 median Filter	PSNR Original VS 480*640	PSNR Original VS 480*640 mean Filter	PSNR Original VS 480*640 median Filter
1	7023.4350	6919.39583	6925.141468	9.665307900	9.730121	9.726517
2	3975.3833	3927.27169	3927.956267	12.13701350	12.18989	12.18913
3	3810.4123	3720.35851	3730.871811	12.32108386	12.42495	12.41270
4	6984.5663	6915.09695	6919.352792	9.689409109	9.732820	9.730148
5	6215.6855	6115.30021	6122.681010	10.19591322	10.26662	10.26138
6	7510.5136	7384.95271	7397.462642	9.374107189	9.447326	9.439975
7	7053.7424	6916.17225	6929.732414	9.646607602	9.732145	9.723638
8	8155.1014	7977.25575	7978.988181	9.016509943	9.112268	9.111325
9	6290.8068	6242.47672	6243.477561	10.14374013	10.17723	10.17653
10	2916.6442	2877.80766	2879.626643	13.48196896	13.54018	13.53744
11	6482.2576	6411.08116	6422.437177	10.01354074	10.06149	10.05380
12	2472.1763	2440.48716	2441.999930	14.20000912	14.25603	14.25334
13	9398.7353	9294.48770	9304.386666	8.400109421	8.448549	8.443926
14	2336.3219	2292.11418	2293.485998	14.44547666	14.52844	14.52584
15	6620.6206	6535.45857	6538.950127	9.921816585	9.978042	9.975723
16	1184.7333	1156.31600	1156.598978	17.39459731	17.50003	17.49897
17	9625.0203	9537.70179	9538.034323	8.296787057	8.336366	8.336214
18	10119.991	9765.48599	9768.370796	8.079002205	8.233864	8.232582
19	9350.0295	9231.83391	9235.766457	8.422673795	8.477923	8.476074
20	6177.5673	6073.36726	6076.776232	10.22262871	10.29650	10.29407
21	10287.716	10207.5605	10209.82137	8.007613611	8.041583	8.040622
22	11092.255	11008.0190	11009.29063	7.680605024	7.713711	7.713210
23	11218.228	11096.6424	11100.13621	7.631560733	7.678887	7.677520
24	10755.836	10659.1637	10662.05357	7.814361581	7.853572	7.852395
25	6614.8622	6507.80180	6508.024109	9.925595567	9.996460	9.996312
26	7685.2892	7276.36841	7289.993297	9.274201423	9.511656	9.503532
27	5413.0152	5234.61596	5235.199986	10.79641106	10.94195	10.94147
28	6314.2175	6201.83187	6202.089042	10.12760823	10.20560	10.20542
29	9678.9436	9571.43459	9572.714367	8.272524001	8.321033	8.320452
30	3672.5764	3566.33706	3568.654794	12.48109512	12.60857	12.60575
31	5814.7439	5753.46705	5755.464902	10.48549765	10.53150	10.52999
32	6421.5941	6370.03981	6371.989671	10.05437504	10.08938	10.08805
33	6390.4681	6107.124966	6116.410922	10.07547687	10.272435	10.265837
34	8852.1706	8745.747444	8750.613356	8.660305855	8.7128342	8.7104186
35	5110.7682	5086.520218	5090.446161	11.04594174	11.066595	11.063245
36	7125.4506	6940.856655	6950.620084	9.602680231	9.7166728	9.7105680
37	5602.9135	5536.000357	5539.282195	10.64666441	10.698842	10.69626
38	9751.7484	9637.652941	9637.159813	8.23997871	8.2910907	8.2913129
39	3487.1802	3405.343433	3409.427772	12.70605963	12.809194	12.803988
40	7755.7666	7697.79141	7700.499187	9.234556264	9.2671422	9.2656148
41	6965.8859	6870.205955	6876.438202	9.701040034	9.7611060	9.7571681
42	4365.7261	4223.818903	4237.384545	11.7302387	11.873750	11.859824
43	6232.4233	6079.641865	6083.14043	10.18423417	10.292023	10.289525
44	4419.9993	4368.474978	4368.678416	11.67658151	11.727505	11.727302
45	5666.3237	5564.65135	5569.965168	10.59778978	10.676424	10.672278
46	7271.7987	7113.484527	7115.801843	9.514385121	9.6099796	9.6085651
47	5776.7533	5724.244792	5723.728523	10.51396533	10.553621	10.554013
48	5186.0400	5149.410391	5150.548343	10.98244497	11.013228	11.012268
49	7698.8304	7604.803648	7601.385224	9.266556064	9.3199235	9.3218761
50	6817.7487	6730.863532	6733.498952	9.794393702	9.8500957	9.848395

No. of video	MSE Original VS 320*480	MSE Original VS 320*480 mean Filter	MSE Original VS 320*480 median Filter	PSNR Original VS 320*480	PSNR Original VS 320*480 mean Filter	PSNR Original VS 320*480 median Filter
1	4515.144	4394.907	4401.1838	11.58409	11.70131	11.69511
2	2852.288	2789.462	2789.6499	13.57887	13.6756	13.67531
3	2565.699	2461.931	2473.2945	14.03875	14.21804	14.19805
4	4589.271	4497.113	4500.38	11.51337	11.60147	11.59831
5	4037.522	3909.007	3915.9469	12.06965	12.21014	12.20244
6	5102.599	4948.953	4958.724	11.05289	11.18567	11.1771
7	4742.565	4579.037	4589.916	11.37067	11.52306	11.51276
8	5792.73	5572.269	5572.1043	10.50197	10.67048	10.67061
9	4323.011	4254.877	4255.2386	11.77294	11.84193	11.84156
10	1772.325	1725.349	1726.2897	15.64537	15.76203	15.75967
11	4152.98	4063.208	4074.9534	11.94721	12.04211	12.02958
12	1595.359	1552.849	1554.2596	16.10222	16.21951	16.21557
13	6441.485	6307.078	6316.5538	10.04094	10.13252	10.126
14	1550.104	1502.017	1503.0284	16.2272	16.36406	16.36113
15	4691.428	4575.045	4578.6367	11.41775	11.52685	11.52344
16	747.2317	714.2511	713.93022	19.39625	19.59229	19.59425
17	6728.97	6602.601	6602.9902	9.851318	9.933653	9.933397
18	7104.386	6824.634	6826.5295	9.615538	9.79001	9.788804
19	6528.531	6364.712	6366.034	9.982649	10.09302	10.09211
20	4099.319	3974.525	3976.8571	12.00369	12.13795	12.1354
21	6886.926	6774.01	6775.367	9.750549	9.822345	9.821475
22	7075.439	6967.804	6967.0522	9.63327	9.699845	9.700313
23	7567.645	7431.534	7433.5341	9.341196	9.420019	9.41885
24	7215.038	7082.481	7083.9195	9.548417	9.628949	9.628067
25	4521.411	4406.452	4406.6841	11.57806	11.68991	11.68968
26	5231.601	4886.939	4894.7552	10.94446	11.24043	11.23349
27	3956.953	3738.654	3738.9954	12.15719	12.40365	12.40325
28	4384.482	4237.596	4237.8509	11.71162	11.85961	11.85935
29	6470.402	6330.749	6331.4321	10.02149	10.11625	10.11578
30	2407.348	2280.193	2281.7952	14.31542	14.55109	14.54804
31	3847.039	3769.218	3770.5672	12.27954	12.36829	12.36674
32	4191.242	4131.233	4133.6673	11.90738	11.97001	11.96745
33	4414.639	4071.552	4075.1814	11.68185	12.0332	12.02933
34	6003.563	5868.954	5872.9035	10.34671	10.4452	10.44227
35	3266.606	3234.099	3238.0496	12.98984	13.03327	13.02797
36	4909.242	4711.152	4717.7839	11.22066	11.39953	11.39342
37	3890.032	3813.929	3815.2438	12.23127	12.31708	12.31558
38	6726.935	6584.18	6583.6843	9.852631	9.945787	9.946114
39	2382.149	2276.207	2279.9905	14.36111	14.55869	14.55147
40	5177.124	5098.262	5100.4979	10.98992	11.05658	11.05468
41	4842.627	4717.356	4721.4469	11.27999	11.39382	11.39005
42	2943.078	2805.022	2813.9764	13.44279	13.65144	13.6376
43	4451.675	4278.554	4280.3149	11.64557	11.81783	11.81605
44	2861.052	2802.43	2803.356	13.56555	13.65546	13.65402
45	4227.671	4107.519	4110.9051	11.86979	11.99501	11.99143
46	5728.694	5520.214	5523.2848	10.55025	10.71124	10.70883
47	3777.721	3713.661	3713.1803	12.3585	12.43278	12.43334
48	3572.161	3531.147	3532.2923	12.60149	12.65165	12.65024
49	5388.768	5276.312	5275.5012	10.81591	10.9075	10.90817
50	4815.997	4700.699	4702.4354	11.30394	11.40918	11.40758

No. of video	MSE Original VS 288*352	MSE Original VS 288*352 mean Filter	MSE Original VS 288*352 median Filter	PSNR Original VS 288*352	PSNR Original VS 288*352 mean Filter	PSNR Original VS 288*352 median Filter
1	8329.208	8173.024	8174.656	8.924766	9.006976	9.006109
2	4626.298	4563.365	4563.153	11.47847	11.53795	11.53815
3	4468.057	4398.157	4403.444	11.62962	11.6981	11.69288
4	8286.931	8184.566	8186.042	8.946866	9.000847	9.000064
5	7522.795	7396.795	7399.306	9.367011	9.440368	9.438894
6	8627.618	8473.805	8478.964	8.771894	8.850019	8.847376
7	8207.745	8066.485	8072.161	8.988565	9.06396	9.060906
8	9253.037	9035.886	9036.746	8.46796	8.571096	8.570683
9	7420.335	7347.5	7348.829	9.426569	9.469408	9.468622
10	3543.019	3490.471	3490.641	12.63707	12.70196	12.70175
11	7637.179	7545.573	7552.365	9.301474	9.353881	9.349974
12	2952.308	2909.92	2910.506	13.42919	13.49199	13.49112
13	10909.67	10766.53	10774.51	7.752688	7.810045	7.806828
14	2787.723	2743.071	2743.366	13.67831	13.74843	13.74797
15	7675.131	7554.75	7557.488	9.279945	9.348603	9.347029
16	1392.018	1355.007	1354.48	16.69436	16.81139	16.81308
17	10670.08	10544	10545.6	7.849125	7.90075	7.900093
18	11123.51	10943.3	10943.46	7.668385	7.739319	7.739258
19	10720.95	10565.07	10566.9	7.828471	7.892078	7.891326
20	7374.524	7238.917	7240.334	9.453464	9.534068	9.533217
21	12013.1	11887.19	11887.7	7.334252	7.380012	7.379826
22	12984.3	12864.07	12864.7	6.996619	7.037019	7.036806
23	12903	12764.41	12764.63	7.023896	7.070795	7.07072
24	12533.17	12395.54	12397.79	7.150193	7.198148	7.19736
25	7998.666	7890.709	7889.778	9.100628	9.159643	9.160156
26	8443.88	8216.197	8216.12	8.865383	8.984095	8.984136
27	5997.386	5793.704	5793.777	10.35118	10.50124	10.50119
28	7333.661	7175.513	7176.309	9.477596	9.572274	9.571792
29	11262.83	11117.36	11117.98	7.614328	7.670787	7.670547
30	4306.249	4178.816	4178.823	11.78981	11.92027	11.92026
31	6874.998	6796.56	6799.082	9.758078	9.807912	9.806301
32	7850.089	7776.316	7776.951	9.182058	9.223065	9.22271
33	7268.558	7039.202	7038.247	9.516321	9.655569	9.656159
34	10325.1	10187.98	10190.04	7.991859	8.049921	8.049045
35	6161.737	6122.404	6124.916	10.23377	10.26158	10.2598
36	8197.01	8006.849	8010.475	8.994249	9.096187	9.094221
37	6428.286	6343.791	6345.002	10.04985	10.10732	10.10649
38	11326.88	11169.76	11169.72	7.589699	7.650365	7.650379
39	3953.946	3842.439	3844.694	12.1605	12.28473	12.28219
40	9276.902	9193.973	9194.096	8.456774	8.495771	8.495713
41	7858.219	7753.509	7759.111	9.177562	9.23582	9.232684
42	4991.504	4849.48	4853.997	11.14849	11.27385	11.26981
43	6856.455	6674.989	6676.065	9.769807	9.886298	9.885598
44	5173.071	5111.175	5111.507	10.99332	11.0456	11.04531
45	6114.044	5987.139	5989.661	10.26752	10.35861	10.35678
46	8041.628	7807.399	7807.881	9.077364	9.20574	9.205472
47	6843.203	6764.639	6763.729	9.778209	9.828357	9.828942
48	5970.969	5922.146	5922.607	10.37036	10.40601	10.40567
49	8948.566	8832.281	8830.568	8.613269	8.670075	8.670917
50	7780.501	7652.833	7654.155	9.220728	9.292581	9.291831

No. of video	MSE Original VS 240*320	MSE Original VS 240*320 mean Filter	MSE Original VS 240*320 median Filter	PSNR Original VS 240*320	PSNR Original VS 240*320 mean Filter	PSNR Original VS 240*320 median Filter
1	6818.437	6640.985	6645.417	9.793955	9.908478	9.905581
2	3916.266	3834.398	3833.971	12.20208	12.29383	12.29432
3	3734.524	3632.988	3640.733	12.40845	12.52816	12.51892
4	6927.967	6807.934	6808.842	9.724745	9.80065	9.800071
5	6096.723	5951.133	5954.147	10.27984	10.38481	10.38261
6	7373.943	7196.144	7200.666	9.453806	9.559805	9.557077
7	6860.671	6704.529	6711.555	9.767138	9.867121	9.862572
8	7917.428	7654.433	7656.152	9.144962	9.291673	9.290698
9	6241.192	6148.624	6149.207	10.17813	10.24302	10.24261
10	2859.524	2799.555	2799.68	13.56787	13.65991	13.65972
11	6411.285	6298.348	6303.9	10.06135	10.13854	10.13471
12	2402.557	2350.396	2351.396	14.32407	14.41939	14.41754
13	9301.332	9129.481	9136.235	8.445352	8.526343	8.523131
14	2265.262	2219.636	2220.649	14.57962	14.66799	14.666
15	6556.874	6411.532	6413.896	9.963836	10.06119	10.05958
16	1145.332	1102.342	1102.041	17.54149	17.70764	17.70883
17	9523.253	9352.902	9352.972	8.342951	8.42134	8.421307
18	9525.107	9302.138	9302.755	8.342105	8.444976	8.444688
19	9248.697	9050.551	9052.465	8.469998	8.564053	8.563135
20	6085.222	5932.354	5933.177	10.28804	10.39853	10.39793
21	10188.09	10030.46	10030.27	8.049877	8.117594	8.117679
22	10970.79	10822.58	10823.24	7.728424	7.787495	7.787232
23	11034.95	10872.66	10876.9	7.703102	7.767447	7.76575
24	10645.23	10474.36	10475.83	7.859253	7.929529	7.92892
25	6401.2	6269.624	6269.31	10.06819	10.15839	10.15861
26	7013.365	6792.638	6793.996	9.671539	9.810419	9.809551
27	5145.25	4922.886	4922.919	11.01674	11.20861	11.20858
28	6179.726	6002.527	6002.906	10.22111	10.34746	10.34719
29	9508.663	9327.904	9328.358	8.349609	8.432963	8.432751
30	3528.757	3392.767	3392.335	12.65459	12.82526	12.82582
31	5719.999	5621.266	5622.112	10.55684	10.63246	10.63181
32	6351.835	6264.569	6265.811	10.10181	10.16189	10.16103
33	6035.879	12934.68	12935.31	6.977245	7.013248	7.013034
34	8765.07	5807.11	5807.138	10.3234	10.4912	10.49118
35	5017.623	8593.318	8595.438	8.70325	8.789195	8.788124
36	6854.841	4963.218	4965.915	11.12582	11.17317	11.17081
37	5420.544	6635.68	6640.177	9.77083	9.911949	9.909007
38	9578.438	5314.992	5315.413	10.79037	10.87578	10.87543
39	3387.271	9388.236	9387.864	8.317857	8.404964	8.405136
40	7673.105	3252.796	3254.712	12.8323	13.00824	13.00568
41	6862.212	7563.682	7565.926	9.281092	9.343471	9.342183
42	4117.621	6722.365	6723.546	9.766163	9.855583	9.85482
43	5982.123	3965.167	3969.201	11.98434	12.14819	12.14377
44	4315.366	5784.822	5785.911	10.36225	10.5079	10.50709
45	5526.394	4236.448	4236.555	11.78063	11.86078	11.86068
46	7091.484	5364.317	5365.793	10.70639	10.83566	10.83446
47	5667.773	6824.913	6824.837	9.623432	9.789832	9.789881
48	5074.007	5568.073	5567.739	10.59668	10.67375	10.67402
49	7622.152	5006.753	5007.374	11.07729	11.13524	11.1347
50	6657.199	7469.269	7468.044	9.310028	9.398023	9.398735

No. of video	MSE Original VS 120*240	MSE Original VS 120*240 mean Filter	MSE Original VS 120*240 median Filter	PSNR Original VS 120*240	PSNR Original VS 120*240 mean Filter	PSNR Original VS 120*240 median Filter
1	110.3409	191.6731	191.1359	27.70344	25.30519	25.31738
2	50.92693	104.9375	104.7277	31.06133	27.92149	27.93019
3	54.90789	142.0816	143.5107	30.73446	26.60542	26.56196
4	54.20314	132.3213	132.5973	30.79056	26.91451	26.90546
5	123.6192	223.3678	226.0203	27.20995	24.6406	24.58933
6	81.87874	204.0186	201.4686	28.99909	25.03411	25.08873
7	121.3944	221.5484	222.4082	27.28882	24.67612	24.6593
8	120.6526	280.5763	280.2707	27.31544	23.65029	23.65503
9	40.4589	110.776	111.2366	32.06066	27.68635	27.66833
10	61.40105	95.7301	95.77236	30.24905	28.32032	28.3184
11	69.63684	136.0893	133.8252	29.70241	26.79257	26.86543
12	105.2392	143.9962	145.0279	27.90903	26.54729	26.51629
13	55.66336	168.8647	167.5641	30.67511	25.85541	25.88899
14	58.92306	86.90698	87.4449	30.42795	28.74026	28.71346
15	62.67817	148.1684	148.3558	30.15964	26.42325	26.41776
16	54.38493	74.62355	75.22448	30.77602	29.40204	29.36721
17	60.9612	166.8585	166.8671	30.28027	25.90732	25.9071
18	228.4366	356.936	356.3327	24.54315	22.6049	22.61225
19	78.7151	189.8777	190.4551	29.17022	25.34606	25.33288
20	175.6349	249.6886	249.7113	25.68469	24.15682	24.15642
21	84.18807	199.8281	199.6484	28.8783	25.12424	25.12815
22	63.19544	166.9075	166.5186	30.12395	25.90604	25.91618
23	67.18953	179.6451	179.6169	29.85779	25.58665	25.58733
24	142.682	252.4213	252.6151	26.58711	24.10954	24.10621
25	147.9987	248.254	247.6477	26.42823	24.18184	24.19246
26	318.0552	485.8954	488.7731	23.10578	21.26538	21.23973
27	135.8505	286.3741	286.3547	26.80019	23.56147	23.56176
28	101.858	204.9128	204.905	28.05085	25.01511	25.01528
29	73.28317	193.5445	193.0314	29.48076	25.26299	25.27452
30	98.32815	154.0189	153.6322	28.20403	26.25506	26.26598
31	58.72855	133.2897	133.1987	30.44231	26.88284	26.8858
32	62.77095	139.3974	140.147	30.15322	26.68826	26.66497
33	291.6337	440.7041	441.4074	23.48243	21.68933	21.68241
34	62.35424	169.0404	168.7173	30.18214	25.8509	25.85921
35	65.71288	126.6493	128.1805	29.9543	27.10478	27.05259
36	181.2225	333.3065	334.2281	25.54868	22.90237	22.89037
37	124.2567	215.677	216.1499	27.1876	24.79277	24.78325
38	78.68741	224.7469	225.1399	29.17175	24.61387	24.60628
39	100.5336	196.351	195.985	28.10769	25.20047	25.20858
40	59.66878	153.4605	153.6718	30.37333	26.27084	26.26486
41	110.9965	200.4689	201.1906	27.67771	25.11033	25.09473
42	124.0266	242.2535	240.2579	27.19565	24.2881	24.32403
43	100.5894	195.1501	193.439	28.10528	25.22712	25.26536
44	90.78376	136.9898	137.0355	28.55072	26.76392	26.76247
45	138.3309	228.056	225.9578	26.72161	24.55039	24.59053
46	197.6829	398.8681	398.7112	25.17111	22.12251	22.12422
47	91.93579	173.5671	173.6556	28.49596	25.73613	25.73392
48	82.98608	147.9403	147.7177	28.94075	26.42994	26.43648
49	46.75583	158.7022	161.2461	31.43245	26.12497	26.05591
50	126.0693	244.7824	245.1535	27.12471	24.243	24.23642

No. of video	MSE Original VS 100*180	MSE Original VS 100*180 mean Filter	MSE Original VS 100*180 median Filter	PSNR Original VS 100*180	PSNR Original VS 100*180 mean Filter	PSNR Original VS 100*180 median Filter
1	169.5264	267.735	267.4074	25.83843	23.85375	23.85907
2	72.48956	136.2326	135.9718	29.52805	26.78799	26.79631
3	73.38513	163.5028	163.0899	29.47472	25.99555	26.00653
4	77.62631	166.5946	166.6144	29.23071	25.91419	25.91368
5	185.2211	282.2412	279.2141	25.4539	23.6246	23.67143
6	128.2671	249.5539	251.0107	27.04965	24.15916	24.13388
7	165.7835	266.07	265.8577	25.93539	23.88084	23.88431
8	175.0817	342.7049	342.4493	25.6984	22.7816	22.78484
9	57.34861	144.1361	144.3061	30.54557	26.54308	26.53796
10	88.28781	126.9612	127.0629	28.6718	27.09409	27.09062
11	101.0426	171.3355	170.9855	28.08576	25.79233	25.80121
12	132.291	173.9691	174.9959	26.9155	25.72608	25.70052
13	83.41328	216.2879	215.1966	28.91845	24.78048	24.80245
14	72.84743	103.29	103.5183	29.50666	27.99022	27.98063
15	93.34913	187.4668	187.5536	28.4297	25.40156	25.39955
16	83.02619	102.3802	102.6194	28.93865	28.02864	28.01851
17	96.09224	218.4454	218.4992	28.30392	24.73737	24.7363
18	169.4533	311.2811	311.1744	25.8403	23.19928	23.20076
19	116.1016	239.5776	239.2944	27.48242	24.33634	24.34148
20	224.2645	307.8402	308.5489	24.6232	23.24755	23.23756
21	126.5063	255.9886	255.6213	27.10968	24.0486	24.05483
22	102.5178	225.4288	225.741	28.02281	24.60071	24.5947
23	92.24726	220.9057	219.8844	28.48127	24.68874	24.70886
24	196.8314	319.7212	319.717	25.18986	23.08309	23.08315
25	184.8292	295.8136	295.3853	25.4631	23.42062	23.42692
26	311.2747	413.98	414.7075	23.19936	21.96101	21.95338
27	166.3901	300.345	300.2944	25.91953	23.3546	23.35533
28	152.1501	264.3771	264.3542	26.30808	23.90856	23.90894
29	120.4875	253.2856	253.3074	27.32138	24.0947	24.09432
30	152.4803	204.793	205.3007	26.29867	25.01765	25.0069
31	88.12681	171.2838	171.3469	28.67972	25.79364	25.79204
32	89.36478	173.9676	175.3518	28.61914	25.72612	25.6917
33	283.5354	358.749	358.8625	23.60473	22.5829	22.58152
34	92.70709	217.6018	217.643	28.45967	24.75418	24.75336
35	96.54204	164.3309	165.7723	28.28364	25.97361	25.93568
36	240.0366	389.3853	390.039	24.32803	22.22701	22.21972
37	158.3439	264.6765	264.855	26.13479	23.90365	23.90072
38	119.4391	277.8071	277.6731	27.35934	23.69337	23.69547
39	146.0179	248.2702	248.438	26.48674	24.18156	24.17862
40	91.269	199.1005	199.7746	28.52757	25.14008	25.1254
41	145.2585	244.5836	244.8928	26.50939	24.24653	24.24104
42	180.9114	293.5978	292.9156	25.55615	23.45328	23.46338
43	153.5281	249.9095	249.1605	26.26893	24.15298	24.16601
44	128.1993	182.054	182.0445	27.05195	25.5288	25.52903
45	182.8213	279.989	279.979	25.51054	23.65939	23.65955
46	265.7172	478.753	478.4235	23.88661	21.32969	21.33268
47	131.4914	222.1935	221.7999	26.94183	24.66349	24.67119
48	104.7841	183.9062	183.8531	27.92785	25.48484	25.48609
49	66.49611	206.0045	210.0261	29.90284	24.99204	24.90807
50	174.5821	308.0604	308.6294	25.71081	23.24444	23.23643

Appendix C: Code

```
%% Main Function
clc
clear
close all


fol1='HQ\';
fol2='480v640\';
fol3='320v480\';
fol4='288v352\';
fol5='240v320\';
fol6='120v240\';
fol7='100v180\';
fol8='C:\Users\Rasheed Sarky\Desktop\25_11_code\code_\80v150\';
fol9='C:\Users\Rasheed Sarky\Desktop\25_11_code\code_\70v120\';



    path_orig=fol1;
    if i==1
        path_diffres=fol2;
    elseif i==2
        path_diffres=fol3;
    elseif i==3
        path_diffres=fol4;
    elseif i==4
        path_diffres=fol5;
    elseif i==5
        path_diffres=fol6;
    elseif i==6
        path_diffres=fol7;
    elseif i==7
        path_diffres=fol8;
    elseif i==8
        path_diffres=fol9;
    end


        filename_orig=[path_orig num2str(1) '.MP4'];
        filename_diffres=[ path_diffres num2str(1) '.MP4'];

        orig_vdo=filename_orig;
        low_vdo=filename_diffres;

% convert video into frames
disp([' video  ' num2str(1) ' is working'])
disp('convert HQ video into frames')
orig_frames=vdo_to_frame(orig_vdo);
disp('convert low res video into frames')
lowres_frames=vdo_to_frame(low_vdo);

disp(['Selected video has ' num2str(size(orig_frames,4)) ' frames'])

 frame_count=2;%input('Enter the number of frames you will use before and after : ');

for i=1:size(orig_frames,4)
    [im,flag]=demo(orig_frames(:,:,:,i)); % function to detect the full face
    if flag==0
        F=i;
        break
    end
end

 if F>=(frame_count+1) && F<(size(orig_frames,4)-frame_count)
        frames_selected_orig=orig_frames(:,:,:,F-frame_count:F+frame_count);
        frames_selected_lowres=lowres_frames(:,:,:,F-frame_count:F+frame_count);
    elseif F<(frame_count+1)
        frames_selected_orig=orig_frames(:,:,:,1:F+frame_count);
        frames_selected_lowres=lowres_frames(:,:,:,1:F+frame_count);
    elseif F>(size(orig_frames,4)-frame_count)
        frames_selected_orig=orig_frames(:,:,:,F-frame_count:end);
        frames_selected_lowres=lowres_frames(:,:,:,F-frame_count:end);
 end
    frames_selected_lowres_N=mean_filter(frames_selected_lowres);
%     frames_selected_lowres_N2=median_filter(frames_selected_lowres);
disp('Comparison between original HQ video frame and low resolution video frame')
[peak_sig_to_noise_ratio, mean_squared_error, max_squared_error, ratio_of_squared_norms]=...
    measerr(imresize(orig_frames(:,:,:,F),[320 480]),imresize(lowres_frames(:,:,:,F),[320 480]))

    disp('----------------------------------------------------------------')
    disp('MEAN OPERATION')
    % newframe_from_orig=mean_of_frames(frames_selected_orig,size(frames_selected_orig,4));
   newframe_from_lowres=mean_of_frames(frames_selected_lowres_N,size(frames_selected_lowres_N,4));
%%here
```

```
    orig_frame=orig_frames(:,:,:,F);

%     figure
%     imshow(uint8(newframe_from_orig))
%     title('Frame from orig frames')
%     saveas(gcf,'Original frames''s super resolution frame.jpg')

%     figure
%     imshow(uint8(newframe_from_lowres))
%     title('Frame from low res frames')
%
%     saveas(gcf,'Low res frames''s super resolution frame.jpg')

%     disp('Comparison between newframe_from_orig and orig_frame')
%     [peak_sig_to_noise_ratio_1, mean_squared_error_1, max_squared_error_1,
ratio_of_squared_norms_1]=...
%         measerr(imresize(newframe_from_orig,[100 100]),imresize(orig_frame,[100 100]));
%
%     ['peak_sig_to_noise_ratio_1  ' 'mean_squared_error_1  ' 'max_squared_error_1  '
'ratio_of_squared_norms_1']
%     [peak_sig_to_noise_ratio_1, mean_squared_error_1, max_squared_error_1,
ratio_of_squared_norms_1]
%
    disp('Comparison between newframe_from_lowres and orig_frame')
    [peak_sig_to_noise_ratio_2, mean_squared_error_2, max_squared_error_2,
ratio_of_squared_norms_2]=...
        measerr(imresize(newframe_from_lowres,[320 480]),imresize(orig_frame,[320 480]));

    ['peak_sig_to_noise_ratio_2  ' 'mean_squared_error_2  ' 'max_squared_error_2  '
'ratio_of_squared_norms_2']
    [peak_sig_to_noise_ratio_2, mean_squared_error_2, max_squared_error_2,
ratio_of_squared_norms_2]


    disp('-----------------------------------------------------------------')
    disp('MEDIAN OPERATION')
   % newframe_from_orig=median_of_frames(frames_selected_orig,size(frames_selected_orig,4));

newframe_from_lowres=median_of_frames(frames_selected_lowres_N,size(frames_selected_lowres_N,4));
%here

%     figure
%     imshow(uint8(newframe_from_orig))
%     title('Frame from orig frames')
%     saveas(gcf,'Original frames''s super resolution frame.jpg')

%     figure
%     imshow(uint8(newframe_from_lowres))
%     title('Frame from low res frames')
%     saveas(gcf,'Low res frames''s super resolution frame.jpg')

%     disp('Comparison between newframe_from_orig and orig_frame')
%
%     [peak_sig_to_noise_ratio_1, mean_squared_error_1, max_squared_error_1,
ratio_of_squared_norms_1]=...
%         measerr(imresize(newframe_from_orig,[100 100]),imresize(orig_frame,[100 100]));
%
%     ['peak_sig_to_noise_ratio_1  ' 'mean_squared_error_1  ' 'max_squared_error_1  '
'ratio_of_squared_norms_1']
%     [peak_sig_to_noise_ratio_1, mean_squared_error_1, max_squared_error_1,
ratio_of_squared_norms_1]
%
    disp('Comparison between newframe_from_lowres and orig_frame')

    [peak_sig_to_noise_ratio_2, mean_squared_error_2, max_squared_error_2,
ratio_of_squared_norms_2]=...
        measerr(imresize(newframe_from_lowres,[320 480]),imresize(orig_frame,[320 480]))  ;

    ['peak_sig_to_noise_ratio_2  ' 'mean_squared_error_2  ' 'max_squared_error_2  '
'ratio_of_squared_norms_2']
    [peak_sig_to_noise_ratio_2, mean_squared_error_2, max_squared_error_2,
ratio_of_squared_norms_2]

%%%
  function frames=vdo_to_frame(filename)

%reading a video file
mov = VideoReader(filename);

%getting no of frames
numFrames = mov.NumberOfFrames;

% number of frames to be used
% numFrames=10;

% creating a data of frames
for t = 1 : numFrames
    currFrame = read(mov, t);
```

```
    frames(:,:,:,t)=currFrame
end
end

%%%

function [im,flag]=demo(img)
flag=0; % if flag=0, that means a full face is detected

detector = buildDetector();
[bbox bbimg faces bbfaces] = detectFaceParts(detector,img,2);

if size(bbfaces,1)==0
    flag=1 ; % if flag=1, that means a full face is not detected, current frame will be dismissed
    im=0;
else
    im=bbfaces{1};
end

%%%

function filtered_frames=mean_filter(frames)

sz=size(frames);
f=sz(end);
for i=1:f
    currframe=frames(:,:,:,i); % one frame from the video
    % resolving the frame in different channels
    currframe1=currframe(:,:,1); % channel 1
    currframe2=currframe(:,:,2); % channel 2
    currframe3=currframe(:,:,3); % channel 3

    % create mean filter
    h = 1/3*ones(3,1);
    H = h*h';

    imfilt_frame1 = filter2(H,currframe1); % apply mean filter on channel 1
    imfilt_frame2 = filter2(H,currframe2); % apply mean filter on channel 2
    imfilt_frame3 = filter2(H,currframe3); % apply mean filter on channel 3

    % saving the filtered variable back to a new variable
    filtered_frames(:,:,1,i)=imfilt_frame1;
    filtered_frames(:,:,2,i)=imfilt_frame2;
    filtered_frames(:,:,3,i)=imfilt_frame3;
end
end

%%%
function filtered_frames=median_filter(frames)

sz=size(frames);
f=sz(end);
for i=1:f
    currframe=frames(:,:,:,i); % one frame from the video
    % resolving the frame in different channels
    currframe1=currframe(:,:,1); % channel 1
    currframe2=currframe(:,:,2); % channel 2
    currframe3=currframe(:,:,3); % channel 3

    imfilt_frame1 = medfilt2(currframe1); % applying median filter on channel 1
    imfilt_frame2 = medfilt2(currframe2); % applying median filter on channel 2
    imfilt_frame3 = medfilt2(currframe3); % applying median filter on channel 3

    % saving the filtered frame into a new variable
    filtered_frames(:,:,1,i)=imfilt_frame1;
    filtered_frames(:,:,2,i)=imfilt_frame2;
    filtered_frames(:,:,3,i)=imfilt_frame3;
end

%%%
% clc
% clear
% close all
%
%
% % reading the video
% [file,path]=uigetfile('*.mp4','Select the original HQ video file');
% vdo=strcat(path,file);
%
% % for vdocount=5
% %     disp('-----------------------------------------------------------------')
% %     disp(['Working on video : ' num2str(vdocount)])
%
%     % select a video
% %     filename = [ num2str(vdocount) '.mp4'];
%     filename=   vdo;
%
%     disp('convert video into frames')
```

```
%     frames=vdo_to_frame(filename);
%
%     disp(['Selected video has ' num2str(size(frames,4)) ' frames'])
%
% %     disp('Input the resolution that you want for video ')
% %     vertical=input('Enter the vertical height of the image: ');
% %     horizontal=input('Enter the horizontal width of the image: ');
%
%     disp('reduce resolution of these frames')
%     for i=1:size(frames,4)
%         lowres_frames(:,:,:,i)=imresize(frames(:,:,:,i),[vertical horizontal]); % restricting
each frame to vertical X horizontal resolution
%     end
%
%     disp('detect frame which have a full face (i.e. 2 eyes, 1 nose and 1 mouth)')
%     for i=1:size(frames,4)
%         [im,flag]=demo(frames(:,:,:,i)); % function to detect the full face
%         if flag==0
%             F=i;
%             break
%         end
%     end
%
%     frame_count=input('Enter the number of frames you will use before and  after: ');
%
%     disp('getting a super resolution frame using the detected faces')
%     if F>=(frame_count+1) && F<(size(frames,4)-frame_count)
%         frames_selected_orig=frames(:,:,:,F-5:F+frame_count);
%         frames_selected_lowres=lowres_frames(:,:,:,F-5:F+frame_count);
%     elseif F<(frame_count+1)
%         frames_selected_orig=frames(:,:,:,1:F+frame_count);
%         frames_selected_lowres=lowres_frames(:,:,:,1:F+frame_count);
%     elseif F>(size(frames,4)-frame_count)
%         frames_selected_orig=frames(:,:,:,F-frame_count:end);
%         frames_selected_lowres=lowres_frames(:,:,:,F-frame_count:end);
%     end
%     disp('-----------------------------------------------------------------')
%     disp('MEAN OPERATION')
%     newframe_from_orig=mean_of_frames(frames_selected_orig,size(frames_selected_orig,4));
%     newframe_from_lowres=mean_of_frames(frames_selected_lowres,size(frames_selected_lowres,4));
%     orig_frame=frames(:,:,:,F);
%
%     figure
%     imshow(uint8(newframe_from_orig))
%     title('Frame from orig frames')
%     saveas(gcf,'Original frames''s super resolution frame.jpg')
%
%     figure
%     imshow(uint8(newframe_from_lowres))
%     title('Frame from low res frames')
%     saveas(gcf,'Low res frames''s super resolution frame.jpg')
%
%     [peak_sig_to_noise_ratio_1, mean_squared_error_1, max_squared_error_1,
ratio_of_squared_norms_1]=...
%         measerr(imresize(newframe_from_orig,[100 100]),imresize(orig_frame,[100 100]));
%
%     ['peak_sig_to_noise_ratio_1  ' 'mean_squared_error_1  ' 'max_squared_error_1  '
'ratio_of_squared_norms_1']
%     [peak_sig_to_noise_ratio_1, mean_squared_error_1, max_squared_error_1,
ratio_of_squared_norms_1]
%
%     [peak_sig_to_noise_ratio_2, mean_squared_error_2, max_squared_error_2,
ratio_of_squared_norms_2]=...
%         measerr(imresize(newframe_from_lowres,[100 100]),imresize(orig_frame,[100 100]));
%
%     ['peak_sig_to_noise_ratio_2  ' 'mean_squared_error_2  ' 'max_squared_error_2  '
'ratio_of_squared_norms_2']
%     [peak_sig_to_noise_ratio_2, mean_squared_error_2, max_squared_error_2,
ratio_of_squared_norms_2]
%
%     disp('-----------------------------------------------------------------')
%     disp('MEDIAN OPERATION')
%     newframe_from_orig=median_of_frames(frames_selected_orig,size(frames_selected_orig,4));
%
newframe_from_lowres=median_of_frames(frames_selected_lowres,size(frames_selected_lowres,4));
%
%     figure
%     imshow(uint8(newframe_from_orig))
%     title('Frame from orig frames')
%     saveas(gcf,'Original frames''s super resolution frame.jpg')
%
%     figure
%     imshow(uint8(newframe_from_lowres))
%     title('Frame from low res frames')
%     saveas(gcf,'Low res frames''s super resolution frame.jpg')
%
%     [peak_sig_to_noise_ratio_1, mean_squared_error_1, max_squared_error_1,
ratio_of_squared_norms_1]=...
```

```
%         measerr(imresize(newframe_from_orig,[100 100]),imresize(orig_frame,[100 100]));
%
%     ['peak_sig_to_noise_ratio_1  ' 'mean_squared_error_1  ' 'max_squared_error_1  '
'ratio_of_squared_norms_1']
%     [peak_sig_to_noise_ratio_1, mean_squared_error_1, max_squared_error_1,
ratio_of_squared_norms_1]
%
%     [peak_sig_to_noise_ratio_2, mean_squared_error_2, max_squared_error_2,
ratio_of_squared_norms_2]=...
%         measerr(imresize(newframe_from_lowres,[100 100]),imresize(orig_frame,[100 100]))  ;
%
%     ['peak_sig_to_noise_ratio_2  ' 'mean_squared_error_2  ' 'max_squared_error_2  '
'ratio_of_squared_norms_2']
%     [peak_sig_to_noise_ratio_2, mean_squared_error_2, max_squared_error_2,
ratio_of_squared_norms_2]
%
%
% % end

function [bbox,bbX,faces,bbfaces] = detectFaceParts(detector,X,thick)

if( nargin < 3 )
 thick = 1;
end

%%%%%%%%%%%%%%%%%%%%%%% detect face %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Detect faces
bbox = step(detector.detector{5}, X);

bbsize = size(bbox);
partsNum = zeros(size(bbox,1),1);

%%%%%%%%%%%%%%%%%%%%%%% detect parts %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nameDetector = {'LeftEye'; 'RightEye'; 'Mouth'; 'Nose'; };
mins = [[12 18]; [12 18]; [15 25]; [15 18]; ];

stdsize = detector.stdsize;

for k=1:4
 if( k == 1 )
  region = [1,int32(stdsize*2/3); 1, int32(stdsize*2/3)];
 elseif( k == 2 )
  region = [int32(stdsize/3),stdsize; 1, int32(stdsize*2/3)];
 elseif( k == 3 )
  region = [1,stdsize; int32(stdsize/3), stdsize];
 elseif( k == 4 )
  region = [int32(stdsize/5),int32(stdsize*4/5); int32(stdsize/3),stdsize];
 else
  region = [1,stdsize;1,stdsize];
 end

 bb = zeros(bbsize);
 for i=1:size(bbox,1)
  XX = X(bbox(i,2):bbox(i,2)+bbox(i,4)-1,bbox(i,1):bbox(i,1)+bbox(i,3)-1,:);
  XX = imresize(XX,[stdsize, stdsize]);
  XX = XX(region(2,1):region(2,2),region(1,1):region(1,2),:);

  b = step(detector.detector{k},XX);

  if( size(b,1) > 0 )
   partsNum(i) = partsNum(i) + 1;

   if( k == 1 )
    b = sortrows(b,1);
   elseif( k == 2 )
    b = flipud(sortrows(b,1));
   elseif( k == 3 )
    b = flipud(sortrows(b,2));
   elseif( k == 4 )
    b = flipud(sortrows(b,3));
   end

   ratio = double(bbox(i,3)) / double(stdsize);
   b(1,1) = int32( ( b(1,1)-1 + region(1,1)-1 ) * ratio + 0.5 ) + bbox(i,1);
   b(1,2) = int32( ( b(1,2)-1 + region(2,1)-1 ) * ratio + 0.5 ) + bbox(i,2);
   b(1,3) = int32( b(1,3) * ratio + 0.5 );
   b(1,4) = int32( b(1,4) * ratio + 0.5 );

   bb(i,:) = b(1,:);
  end
 end
 bbox = [bbox,bb];

 p = ( sum(bb') == 0 );
 bb(p,:) = [];
end
```

```
%%%%%%%%%%%%%%%%%%%%%%% draw faces %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
bbox = [bbox,partsNum];
bbox(partsNum<=2,:)=[];

if( thick >= 0 )
 t = (thick-1)/2;
 t0 = -int32(ceil(t));
 t1 = int32(floor(t));
else
 t0 = 0;
 t1 = 0;
end

bbX = X;

%%%%%%%%%%%%%%%%%%%%%%% faces %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if( nargout > 2 )
 faces = cell(size(bbox,1),1);
 bbfaces = cell(size(bbox,1),1);
 for i=1:size(bbox,1)
  faces{i,1} = X(bbox(i,2):bbox(i,2)+bbox(i,4)-1,bbox(i,1):bbox(i,1)+bbox(i,3)-1,:);
  bbfaces{i,1} = bbX(bbox(i,2):bbox(i,2)+bbox(i,4)-1,bbox(i,1):bbox(i,1)+bbox(i,3)-1,:);
 end
end

%%%
```