Pratyush

Ghosh

September 2020

Detection of Moving Objects and their tracking in Low-Quality CCTV recorded footage(s) with remote alerting and noise detection using a secured DWT based blind watermark

Master of Science (M.Sc. Electrical and Electronics Engineering)

First Supervisor: Dr. Ming Hou

Second Marker: Dr. Ian Bell

THE UNIVERSITY OF HULL

THE UNIVERSITY OF HULL

Detection of Moving Objects and their tracking in Low-Quality CCTV recorded footage(s) with remote alerting and noise detection using a secured DWT based blind **watermark**

being a Dissertation submitted in partial fulfillment of

the requirements for the Degree of

MSc **Electrical and Electronics Engineering**

in the University of Hull

By

Pratyush Ghosh

Abstract

The report is a bulk action consisting of three consecutive sequences, this has been done keeping the various terms and principles of video and image processing in mind as well as ensuring the noise produced is to its absolute minimum. The sequences are **detection and tracking of a moving object**, **video quality enhancement, and video watermarking using Discrete Wavelet Transform (series of image(s) watermarking**).

For the **detection and tracking of a moving object,** a background model (single- mode) based on a blob-analysis is utilized to separate the foreground from the sequences of images (video frames) in a complex or simple environment. First, the symmetric difference is used which roughly helps to extract a moving object. Then blob-analysis comes into play, this analysis is used to update the background model accordingly. Then at last the final stage in this sequence is the classification strategy (generally frame level) this is needed to extract the foreground properly and as much as possible to the best of efforts to reduce the effect of noise and variance of the illumination. The concept of video surveillance is widely used for monitoring key areas such as banks, traffic stops, commercial stores or building complexes, etc. with the help of sophisticated technology in hardware and software systems it has proved quite helpful in storing large volumes of data and also processing them with fairly good speeds. Earlier the video surveillance was done manually and was saved in tapes to be "looked at" later. However, with the increasing number of cameras in the surveillance sector, it is quite difficult to ensure proper "surveillance" of every video manually without making some delays or errors. To help reduce the manual labor and increase the accuracy and efficiency/response time of the said videos for detecting a moving object, for tracking them in specific events in both live and recorded videos, moving object detection has become quite critical in the modern age. This specific report will use the recorded video from a camera and process it to detect the objects (foreground) from any available video format (meaning color or grayscale). A tracking algorithm will be used to track the "detected object".

For **video quality enhancement,** many image(s) enhancement algorithms have been used in various applications. Despite that, some algorithms and modules have proved to be quite imperfect when it comes to practical application. This report aims to find a useful algorithm to ensure the low-quality image(s) (frame) enhancement modules. A brief review of some enhancement methods will be briefly mentioned and the paper will also mention the algorithm used in this report. the results will be described and discussed as well.

For **video watermarking using Discrete Wavelet Transform (series of image(s) watermarking**), it (video/image watermarking) is believed to be one of the most important and powerful tools when it comes to ownership, copyrights protection in digital media. This report proposes a "blind digital watermarking (video) technique based on a combination of **"Discrete Wavelet Transform"** and the **"real Schur Decomposition".** The algorithm will start by applying a two-level Discrete Wavelet Transform to the video sequence which will then be followed by Schur Decomposition where the binary watermark bits will be embedded in the block of the upper triangular matrix. The proposed algorithm seems to show a high level of efficiency solely because of the incorporation of Schur Decomposition which needs lesser computations and calculations and still gives a good result when compared to other transform methods. With the added use of the Discrete Wavelet transform the imperceptibility of the algorithm is quite high, as a result, there is no visual distortion noticed in the watermarked video after the embedding procedure. Moreover, the proposed algorithm and technique is proved to be very robust against a bunch of standard attacks such as Gaussian, Salt and pepper noise, rotation, and against video attacks such as cropping, frame dropping, and averaging. The capacity and blindness features are also considered and are achieved using this method.

Acknowledgments

Throughout the writing of this dissertation, I have received a great deal of support and assistance. I would first like to thank my main supervisor, **Dr. Ming Hou**, as well as my second project supervisor **Dr. Ian Bell,** whose expertise was invaluable and critical in the formulating of the research topic and methodology without whom this would not be possible.

I would like to acknowledge my M.Sc. Individual Project module leader **Dr. Ian Bell**; I want to thank you for your excellent cooperation and for all the opportunities I was given to conduct my research and further my dissertation at **The University of Hull**. I would also like to thank my M.Sc. personal supervisor, **Dr. Anthony Wilkinson**, for his valuable guidance. You provided me with the tools that I needed to choose the right direction and complete my dissertation.

Also, I would like to thank my parents for their wise counsel and sympathetic ear. You are always there for me. Finally, there are my friends, who were of great support in deliberating over the problems and findings, as well as providing a happy distraction to rest my mind outside of my research.

Table of Contents

Table of Figures

Chapter 1: Introduction

The computer vision systems have been developed to process and simulate most of the systems (biological and non-biological) with different environmental conditions. Simply put the detection of moving objects and then tracking the said objects can be seen as a lower-level vision task that can and will be needed to achieve a higher-level event understanding. Identifying the moving objects is a very important task for video segmentation, this has been used by many computer vision applications like video surveillance, remote sensing, traffic monitoring, etc. recently, video analysis has found importance in many real-time monitoring applications. It has found applications in security fields such as video surveillance in banks, military unauthorized zones, and simple CCTV cameras in stores. Before image or video processing found any advancement in its applications there were a lot of restrictions to perform complex level video processing (real-time as well as recorded).

Recently a few advancements have helped improve video processing some of these advancements are rapid computation, high definition cameras, and improvements to the algorithm and code which has helped people overcome a lot of the limitations caused by traditional technologies. Analyzing a video consists of two main processes

1) **Object Detection**
2) **Object Tracking**

For the case of **Object Detection,** it helps identify the region of interest (segment) which is commonly called the foreground object. This algorithm uses blob-analysis as the main engine for detecting, the background subtraction method is needed to segregate the reference model and the region of interest. Over here the result of blob analysis is used as in input to a morphological algorithm, which will help in improving accuracy by removing unnecessary pixels which might be noise. Then the tracking algorithm is enabled to track the detected objects by their change in pixels direction after every frame. In this algorithm the processed video will be saved and in this and the upcoming enhancement and watermarking algorithms will contain a **save to hardware function** and **remote alerting via electronic mail** included.

Image/video processing has been considered very important and have been developed accordingly and is regarded as an important research field currently. It is also in great demand in various fields namely biology, archaeology, medicine, aeronautic applications, and display industries to name a few. As stated previously there are countless image/video processing algorithms used for the enhancement, but usually, these algorithms are somewhat imperfect. Amazing results could still be obtained if the tuning would be manually handled. In practical applications, the module for processing has been made to work either alone or as an integrated algorithm. The primary aim of this report is to compare three embodiments and find out the most suitable and proper result giving method. The applications are **artifacts reduction, resolution upscaling, deblurring**. Through trial and error, and inability to switch to other software(s) it was seen that **deblurring** was a better choice the reason for this choice and explanation of other methods will be explained in **Chapter 6: Design Ideas and Concept Selection**

Due to the development of blogs, video sharing websites, and social network applications, the term digital videos and images are remarkably a necessity in the modern world. Due to its increased use and importance, it becomes necessary to protect the copyrights of the digital content guaranteeing ownership identification. In multimedia digital watermarking has proven to be the most efficient tool in securing data (digital). Practically the watermarking method embeds a small copyright message or logo which is called a watermark within the digital media which will be unique to a person or company or team. Highly effective techniques in watermarking will have to contain robustness and imperceptibility. Imperceptibly is also called watermark perceptual transparency (Al-Khatib, 2008) of the algorithm, while robustness is the ability to extract the embedded watermark from the digital media under attacks (intentional and unintentional) like degradation, cropping, filtering, and/or additive noise (Al-Khatib, 2008) (Voloshynovskyy, 2001). Some issues such as capacity and blindness must also be considered as well. Capacity is referred to as the data payload which includes methods that make it possible in embedding most information of course in certain formats. The blindness of any algorithm means that the extraction after embedding a watermark does not require the presence of the original image (Baisa L Gunjal, 2010). Here as well there have been numerous watermarking algorithms that have been proposed for video, video frames (images), and image(s). generally, digital watermarking methods are classified into two groups, spatial domain watermarking used in (Rosa Lancini, 2002) (Hussein, Jan 2010) (Mondal, Jan 2012), and transform domain watermarking which are used in (Salwa A.K Mostafa, 2009) (Sadik Ali M. Al-Taweel, 2009) (Sathik, 2012) (M.C. Hernandez, 2005).

In the spatial domain method, the watermarking is done by changing/modifying the pixel values present in the host image. Whereas in the transform domain watermarking method, the watermark is embedded in the transformed coefficients of the image. The transform domain watermarking methods (commonly known as frequency-domain methods) are seen to be more robust and complicated than the spatial domain watermarking method, and they are visually more of a convenience to people and have a high imperceptibility and capacity (Lama Rajab, 2009/8).

To guarantee ownership and copyrights identification the digital media needs to be protected, and so watermarking can be applied to these media (videos, audios, images, and documents). For digital videos watermarking can be done on both compressed and uncompressed formats. However, it is better and safer to apply watermarking on un-compressed formats because the frames of the uncompressed video are consecutive and equally spaced, so the watermark image will be embedded without any risk of losing the data. However, the watermarked compressed video such as MPEG formats (compressed bitstream) can be converted into a different compression standard which can result in either medication or ultimately losing the embedded data (Lama Rajab, 2008). Currently, many researchers are working on transforming domain watermarking for media such as audio, images, compressed and uncompressed video formats.

Various algorithms were proposed that made use of common transformations such as SVD, DCT, and DWT. Still, many algorithms combined some of these available transformations and created hybrid approaches solely to obtain a better image (watermark) embedding an image (watermark) extraction results. In some other researches, the SCHUR transform was considered and studied. Using this transform (SCHUR) in watermarking embedding and extraction, it was seen that it was quite promising and eventually would compete with other transformations and give great results when it came to topics such as robustness and precision in watermarking. In (K Meenakshi, 2014). There was a discussion that presented a hybrid non-blind watermarking algorithm using SCHUR and SVD transform(s).

in this, SCHUR in embedding was used in embedding and extraction a few steps after the image was decomposed into 8 x 8 blocks, then followed by applying SVD on the largest Eigenvalues. But, in (Panahi, 2013), there was a proposal for a non-blind hybrid method utilizing SCHUR and Cellular Automata transform (CAT) which used the flexibility of the CAT domain and the efficiency of the SCHUR transform. While in (Hassen Seddik, 2009). A blind image watermarking method solely based on the SCHUR transform was proposed. The image is transformed into SCHUR transform and then analyzed to get a non-sensitive location (zone) where the embedding procedure takes place. Whereas, on the other hand, the proposed algorithm introduced in (Swamy, 2010), embeds the watermark into two decompositions of the SCHUR transform, they are the D and the U matrix. This technique improves the robustness of the algorithm since the same watermark data has been stored twice in two places.

Chapter 2: Review of Previous Work

In today's world applications of software are quite common, namely in the fields of image or video processing.

There are many programmers out there who something has also done like what will be covered in the coming topics. More precisely, YES, the three subparts of this report have been done elsewhere by different companies and/or programmers. Then what's so different in this report? What makes this project slightly standout?

At present many research topics focus solely on either object detection and tracking (for moving objects), either a video (series of frames) quality enhancement, or either Image(s) watermarking through countless algorithms and methods. Very rarely has a report been encountered combining all three of the above mentioned. The reason may be possible because each of the above-mentioned methods involves the hardware processor reaching its peak limit and the possibility of combining all the three algorithms could risk overheating the hardware and perhaps crashing the software. One possible outcome through trial and error was that if each algorithm for each of the subparts was chosen in such a way that the hardware would have to perform fewer computations and give a decent amount of fewer noise results than any other high calculation algorithm. Because as even little noise could give different results and in applications such as government defense, military, police these errors could lead to extensive damages.

To counter the noise factor with the least computations for the hardware the report makes use of the morphology function that helps eliminate the noise with four to five options depending on the user and the scenario, without destroying any data. another method to help reduce the computations with the help of trial and error was by saving the output videos in each of the sub-section in a specific format.

Other research reports commonly save the output video in MPEG-4 format, also known as MP4. The MP4 format is a highly compressed format (lossy) which is popular in today's world because it takes less space and gives a good enough performance. Due to this compression to save space certain details in the video are compromised and although it may not appear to the naked eye it does appear as noise in the video processing algorithms. This report saved the output in AVI formats. The AVI file as stated uses less compression than other formats and hence results in the output file being stored with a larger size as compared to MPEG and/or MOV files. AVI formats can also be created without compression (use in this project) making them lossless again resulting in much larger file size.

A lossless file unlike the lossy (compressed) format will not lose its quality regardless of the number of times the file is opened or viewed or saved. By this simple conversion, it is possible to combine all three methods and give an output with less noise as well as detected objects, enhanced frames, and additional security through the image(s) watermarking (GAVIN, 2018) .

In addition to the above facts, the codes present in this report have remote alerting via E-Mail which will alert the user once the code has run and can be modified to attach the compressed output or the various enhanced frames as well as the graph for showing the successful or unsuccessful embedding and extraction of the watermark in the image frames. The Entire Code, the code subparts, and some of the extra results will be shown in Appendix A and Appendix B.

Chapter 3: Ethical Issues

I am a member of a globally renowned University and am directly/indirectly responsible for the improvement of fields such as technology and innovation, which in turn will affect the quality of life worldwide. In accepting a personal obligation to my role as an M.Sc. Student and a researcher, I hereby commit myself to the highest Ethical (Principles of research ethics, n.d.) and professional Code and Conduct.

1. To accept the responsibility in all research related decision making, keeping in mind the health and safety as well as the welfare of the public.

2. To be Honest and Fair to the best of my knowledge.

3. To strive for excellence in the Engineering Department and enhance my knowledge and skills without stealing other's property and time.

4. To avoid intentional (real) and/or unintentional (perceived) conflicts of interest whenever and wherever possible.

5. If given the chance to present my skills and understanding of the topic via projects and/or educational courses/programs, and not hesitate to seek help when faced with a situation beyond my abilities and understanding.

6. To respect and understand the professional advice from the supervisors and learn from their mistakes and experience.

7. To treat this report as a chance to learn new methods and respond professionally accordingly.

8. To give a strong rejection of bribery, mal-practice, and any form of professional misconduct in all forms.

9. To convey the knowledge and skills gained to others, to help everyone gain the fruit of each other’s knowledge.

10. To understand the technology (appropriate application, consequences) and help in its improvement.

11. To accept the honest feedback or criticism of the work, to understand and correct the errors.

12. To treat all persons in the department fairly, irrespective of sex, race, religion, age, disability, origin, and not destroy the reputation by false documents.

13. To differentiate science from advocacy and not present ideas and analysis as a universal truth

14. To respect others, work and give them the due credit. Never steal or misuse copyrighted material.

Chapter 4: Specifications

This project will not require any hardware (except for the case of future scopes, and the laptop/computer this code will run on),

Amongst the software requirements, this project will need an up-to-date MATLAB package installed. More specifically MATLAB along with

- Math, statistics, and optimization Toolbox
- Automotive Toolbox
- Signal Processing and Communication Toolbox
- Image Processing and Computer vision Toolbox
- Machine Learning and Deep Learning Toolbox

For certain functions and coding instructions to work properly.

This report was programmed and run on hardware having a dual-core processor. Other than this there will be no hardware used.

Chapter 5: Explanation (description) of Methods and Procedures

1) Detection and Tracking of a Moving Object

1.1) Object Detection Only

The Object Detection method involves differentiating the foreground objects from the still (stationary) background. The first step taken is plainly to detect a foreground object. Then a base will be made for further operations such as tracking and classification (sometimes when CNN can be used). Initially, the background scene initialization is performed. In this report, the foreground detector (function) can help differentiate the background. Further, the foreground pixel will be detected from the background model and the current input frame from the video. Before further processing is done pixel-level post-processing occurs to help remove the noise in the foreground pixel (this report has utilized the morphological operators with different options to reduce the noise pixels). The foreground detection, background detection, and pixel-level post-processing (morphology) are mixed to create an object pixel map in the foreground and receive the objects in the video frame.

1.2) Object Tracking

This method is quite complex and difficult to fully understand, therefore getting the attention and gaining the interest of many researchers globally. The primary aim in this report specifically is to form a correlation of objects and various segments of the said objects within the consecutive frames (video). The object tracking idea is very important in the process of moving object detection. It has its uses and relative importance in surveillance applications because it can give temporal data regarding the moving elements within the frames. The algorithm for object tracking utilizes aspects of the extracted objects with a corresponding matching to track the very objects of interest from one frame to another.

Using the object tracking algorithm in a cluttered environment is quite tricky, the reason being the segmentation problems such as wide and narrow shadows cast by the objects, complete or partial obstruction of the objects specifically with each other, the stationary objects in the environment (parked cars, trees, billboards) for a still background. The main purpose of the object tracking algorithm is to get temporal information of objects such as the position of the object, path trajectory (best if used using the KLT algorithm), the direction of the moving object, and its speed covered for a given distance.

There are two methods in tracking the moving objects,

1) Tracking the moving objects using motion prediction and/or position estimation
2) Track different parts of the object differently

This report will focus on using object level tracking, which tracks the object from one frame to another. A flow diagram of what will be done in this segment of the report (**detection and tracking of a moving object**) Is shown in Figure 1.

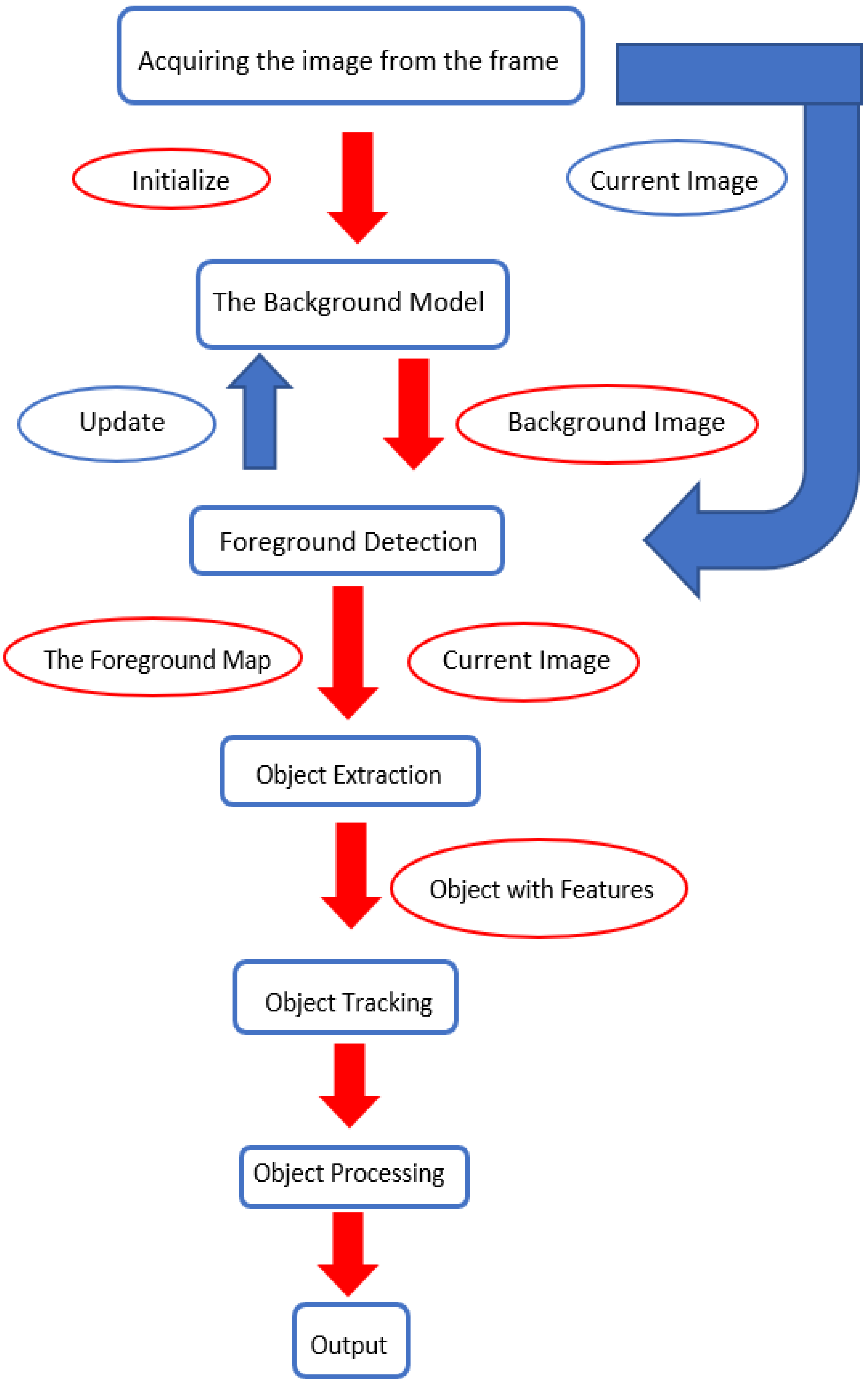


Figure 1 Flow Diagram for the Proposed Object Detection and Tracking Algorithm

1.3) Background Subtraction

The input videos in this report contain a static background, meaning the device that recorded these videos were fixed in a specific spot. This is the main motivation in utilizing the background subtraction approach. The background subtraction approach is one of the popular methods that has found applications regarding the detection of moving objects. Amongst other approaches, it is considered to be the most efficient technique.

The use of the background subtraction process is to leave only the foreground pixels of the interesting objects (moving) by removing the background pixels. The comparison for this algorithm is made between the i^{th} frame and the $(i-1)^{th}$ frame. The background subtraction method matches the reference model with the first number of frames of the input video. The intensity values of the pixels in a given frame will then be subtracted with the pixels of the reference model. Furthermore, the difference in the pixel values is then filtered using the threshold per pixel (Dipali Shahare, 2014) in equations (1), (2), and (3).

D(x,y) = $|\mathbf{It(x,y) - Bt(x,y)}|$**………………………………………………………………………………… (1)**

D(x,y) > th = 1……………………………………………………………………………………………… (2)

= 0, otherwise ………………………………………………………………………………………………… (3)

In a set of equations (1), (2), and (3), **It(x,y)** is the intensity value of the pixel at the coordinates **(x,y)**. For the video sequence **I** and the respective time frame **t**. the term **Bt(x,y)** is the respective background pixel value. The th term being the threshold value (approximate) through the image sequence. The reference model Bt(x,y) is initialized with the first video image (frame) $\boldsymbol{I_0}$, $\boldsymbol{B_0}$ **=** $\boldsymbol{I_0}$**.** The threshold will be defined using a pre-determined value. a pixel will be set as a background pixel if **D(x,y) < th**.

Several holes will be generated in the binary mask of the moving object due to the similar pixels that are present in both the background and the foreground. To solve the issue of the hollow phenomenon, dilation and/or erosion of the morphological operators (Rafael C. Gonzalez, 2002), is used. It not only will fill up the hollow but also help in removing the noise of the binary image. It can therefore provide a complete mask of that object moving for obtaining a much better extraction later. A small example is shown in Figure 2 and Figure 3. The resulting background subtraction modules of the other 2 videos used will be present in Appendix A.

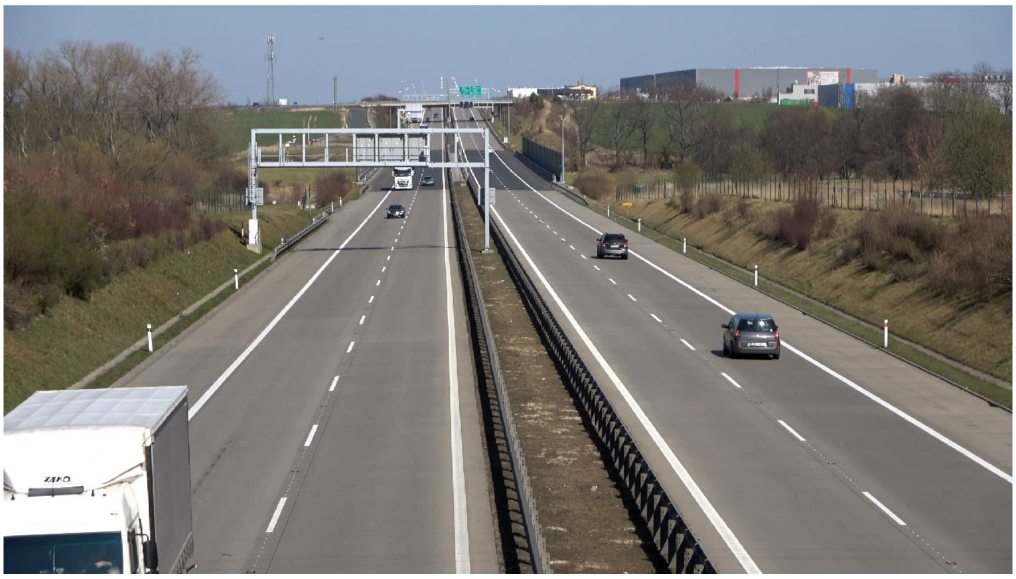


Figure 2 Original Frame of the Sample Video 1 used

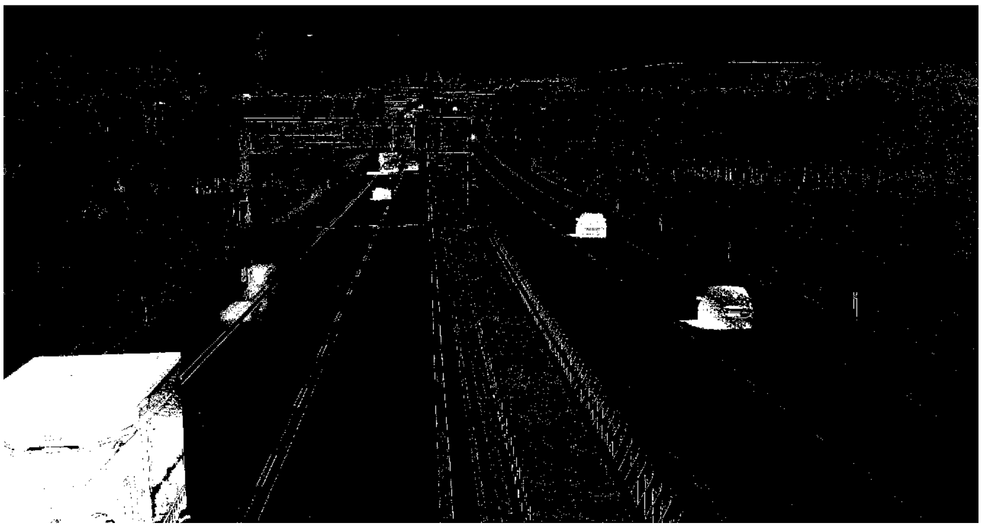


Figure 3 Result of the Sample Video 1 frame after Background Subtraction Module (with some noise)

1.4) Frame Difference

The background image [please refer equation (4)] is obtained from the frame **(f)** which exists before the current video frame f_{t-1} (Hario Baskoro Basoeki, 2016).

B = f_{t-1} ………………………………………………………………………………………………… **(4)**

1.5) Weighted Moving Mean

The mean of each frame is computed using the algorithm (Hario Baskoro Basoeki, 2016), in equations (5) and (6)

B = $(f_t * w_1) + (f_{t-1} * w_2) + (f_{t-2} * w_3)$ ……………………………………………………… **(5)**

B = $((1/t)\sum_{i=1}^{t}(f_i - (f_i * w_1) + (f_{i-1} * w_2) + (f_{i-2} * w_3)))^{1/2}$ ……………………… **(6)**

1.6) Weighted Moving Variance

The variance of each frame of the video will be calculated. The frame difference, the weighted moving mean algorithm as well as the weighted moving variance algorithm will be used for creating the reference model of the background. To detect the objects from the foreground model, it is necessary to calculate the difference between the reference model and the current image (frame). The equation for calculating the variance is (Hario Baskoro Basoeki, 2016) shown in equation (7)

F = $|f_t - \mathrm{B}|$ ………………………………………………………………………………………………… **(7)**

1.7) Single Gaussian

A Gaussian model is used for calculating the values of each pixel from the mean (μ) and the variance (σ) of the sample pixels. Following that a lower and upper bound value is set which will be used to eliminate the pixels which will not follow the constraints. To adapt the model over a period at the new frame, the mean [equation (9)] and the variance [equation (8)] must be updated by taking the average of the pixel values (Hario Baskoro Basoeki, 2016).

$$\sigma(x,y,t) = \frac{\left(\sum_{i=1}^{t}\frac{p_2(x,y,i)}{t} - \mu_2(x,y,t)\right)}{2} \quad \text{.......... (8)}$$

$$\mu(x,y,t) = \sum_{i=1}^{t}\frac{p_2(x,y,i)}{t} \quad \text{.......... (9)}$$

Detecting the foreground object is done by the equation (10),

$$|\mu(t+1) - \sigma(t+1)| < k\,(\sigma(t+1)) \quad \text{.......... (10)}$$

The pixel will then be classified as a foreground or a background pixel.

1.8) <u>Morphological Operation</u>

After the completion of the background subtraction, the foreground mask can be obtained which will represent the moving pixels within the video frame, which unfortunately can contain noise as seen in Figure 3. To overcome this problem the morphological operations can be used to remove most of the noise (Hario Baskoro Basoeki, 2016). Some of the morphological operations are shown in Figure 4, Figure 5, Figure 6.

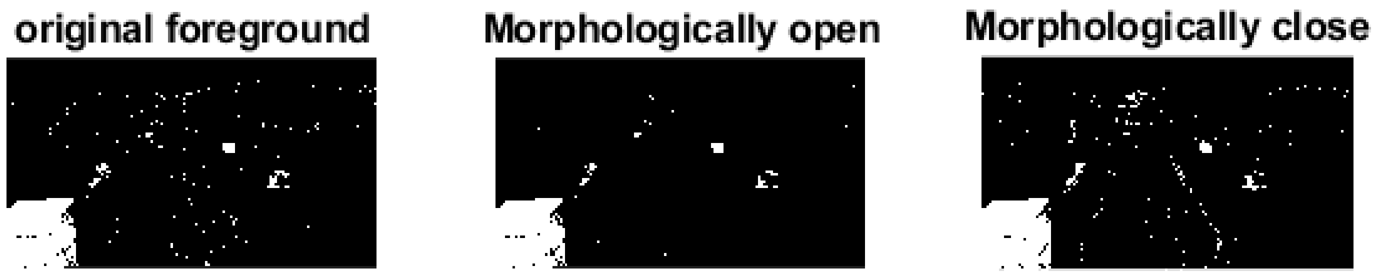


Figure 4 The Original Foreground from Figure 3 (left), the Open Morphological Operation (middle), the Close Morphological Operation (right)

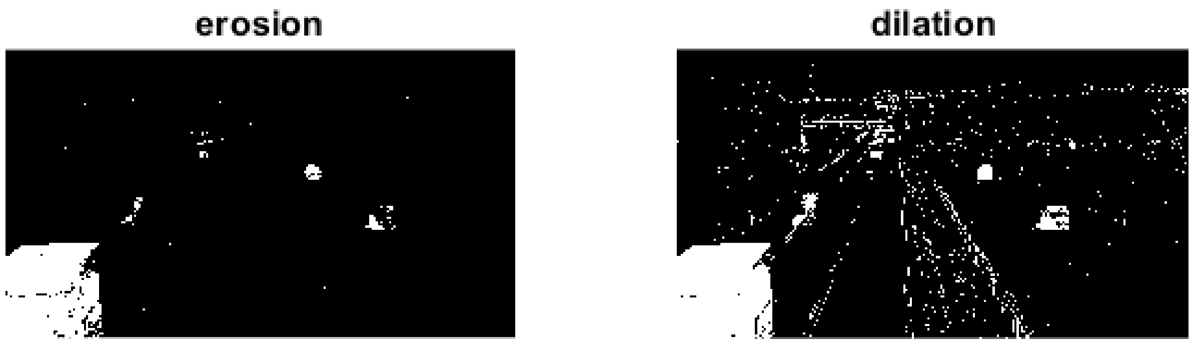


Figure 5 The Erode Morphological Operation (left), the Dilate Morphological Operation (right)

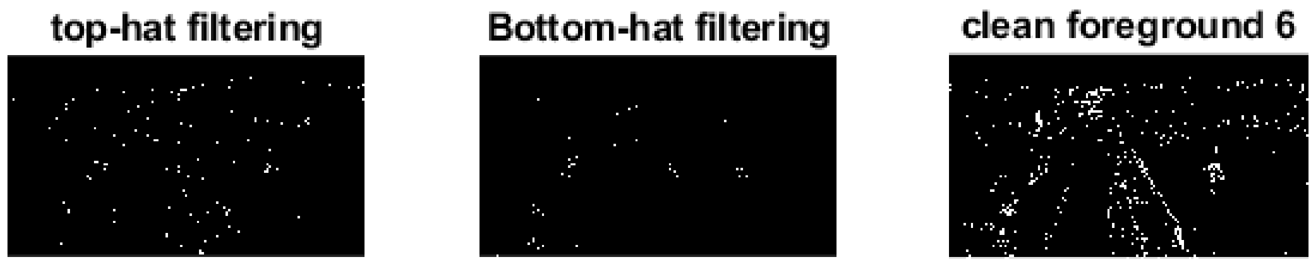


Figure 6 The Top-hat Morphological Operation (left), the Bottom-hat Morphological Operation (middle), the Morphological Operation obtained by subtracting Dilate and Erode (from Figure 5)

In this report the decision has been made to proceed with the erode morphological operation (Figure 5), as based solely on eye judgment it does a great job at reducing most of the noise as well as managing to conserve or the object pixels by not removing their data. The morphological operators for the other 2 videos that will also be used will be present in Appendix A.

1.9) Blob Analysis

The blob is considered as a part of the image where some aspects are partially constant. Blob analysis is being used in this report due to its high flexibility and good performance. Once the foreground object has been detected, a term called blob detection (explained in chapter 6) can be used to increase the efficiency of the detected region.

Blob Analysis uses the three-step method that is,

- Extraction
- Processing
- Analysis

In the first step, the thresholding technique is applied and either the region of the objects or the objects itself will be detected. A lot of times the detected or the extracted regions will contain some noise. In the next step, the objects or the regions are updated using a transformation method. In the analysis stage, the detected objects will be processed and split into many blobs so that they can be recognized separately (Tao Jia, 2008).

2) Video Quality Enhancement

2.1) Enhancement of Image(s)

Earlier the Digital Image processing methods were needed in the newspaper industry and also the early 20th-century space project, which has grown quite rapidly due to the development of digital computers, high end backing up devices to store data, the transmission of data and display. It has however found applications in a wide range in the past as well, such as blurred pictures that were sharpened by an image processing method in archaeology. The degraded images were repaired using image enhancement methods in geography (Philippe Thevenaz, 2000).

Digital Image processing itself constitutes many aspects such as Image Enhancement, Compression, Image Acquisition, Image Restoration, etc (Philippe Thevenaz, 2000). This report will focus on the basic deblurring algorithm only. Other potential methods will be explained in chapter 6.

I. Deblurring

Deblurring is also known as sharpness enhancement. Sharpening is the opposite operation of the blurring process. In the Spatial domain, during blurring the averaging method can be used. For sharpening the mathematical models can also be used to do the differential calculations.

Commonly the detail-sharpening filters (spatial) are based on the first and second derivatives, for examples such as the Laplacian (using the second derivative) (P. Thévenaz, 2000) and the Gradient (using the first derivative) (P. Thévenaz, 2000) . In the frequency domain, the high-pass filter is used to help weaken the low-frequency content and at the same time protect the high-frequency content in the Fourier transform.

They are generally of 4 types, Idea Butterworth, Laplacian (P. Thévenaz, 2000) & Gaussian. There are also a few advanced methods such as high-boost filtering and unsharp masking that are used by high-end devices. Figure 7 shows a result produced by using the unsharp masking technique.

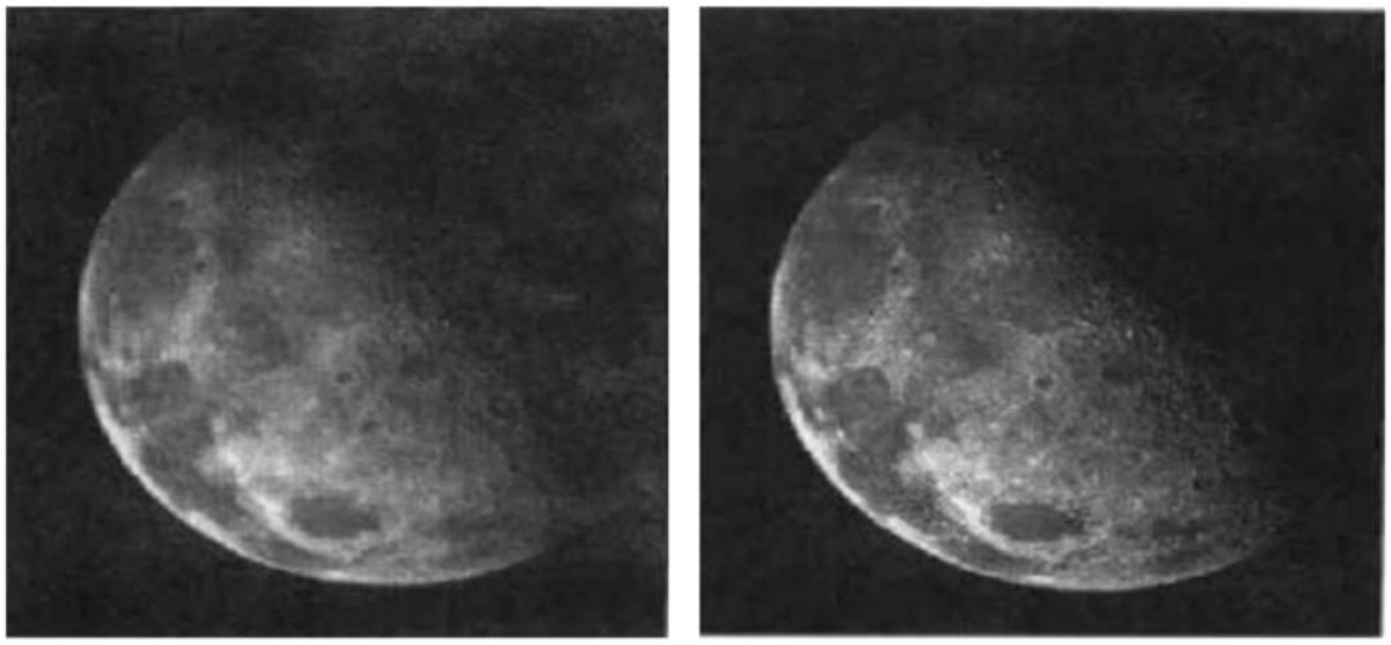


Figure 7 The Image (right) is the Original, the Image (left) is the result of the unsharp masking method (P. Thévenaz, 2000)

2.2) Least Mean Square Optimization

I. Solving Least Squares Problem

The Least Square Algorithm is said to be the standard approach to use partial derivatives which will be used to get the conditions necessary for a minimum, and then to solve the equations involved (Aggarwal, 2010). This algorithm is generally used in data fitting situations including (but not limited to) approximate data model, extrapolation, and data smoothing. This report, however, focuses on the methods to solve the matrix which will be used in the following subsection (least mean square filter). There will be 4 possible methods which will help solve this problem. They are the QR Transformation, the Gaussian elimination, the Conjugate gradient, and the singular value decomposition (also known as SVD).

The Gaussian elimination method might be faster than the QR Transformation given the condition the matrix is not square. However, the QR transformation which uses orthogonal decomposition is much more stable. The Conjugate gradient will be a better choice for the larger sparse problem, whilst the SVD will only indicate how independent the columns are.

II. Least Mean Squares Filter

The LMS filter aims at reducing the mean square error (the MSE) between the ideal and the targeted image. Amongst various other video enhancement algorithms, the least mean square filtering method is known to give the most impressive results, including the Kondo's Classification based Least Mean Squares (Yovits, 1980) filter algorithm, then the Li's Localized Least mean Square filter algorithm (Li & Orchard, 2001) on some of the resolution-up-conversion problems.

The Least Mean Square Optimization algorithm has a strong theoretical background and also has a big advantage where the parameters can usually be calculated from a degraded image (P. Thévenaz, 2000).

2.3) The Pixels Classification Method

Many classification methods have been used to "classify" the pixels in an image as per the many different pixel properties, like color, intensity(grey-level), and/or the local image characteristics based on the pixels neighborhoods (US Patent No. US5444487A, 1995). Using the method, the difference between each class is much more valuable as compared to the disparity in the same class. This report the classification can be used to distinguish the local structures and the complexity, these are the two very important properties that are used in any image(s) enhancement algorithm.

I. Classification on Basis Structures

To the image quality, an Adaptive Dynamic Range Coding (also called ADRC) (United States Patent No. US6483546B1, 2002), proposed by T.Kondo & K.Kawaguchi can be used to possess a high-efficiency fore simply representing the local structure. In practical applications, the brightness ranges from 0 to 255 as the value of a specific pixel can be calculated for the classification. First, the images are prepared by dividing the blocks in a specific size. Each pixel will then be encoded into the image into a 1-bit with ARDC, more specifically the ARDC code of each pixel. $\boldsymbol{Q_i}$ in an observation block is defined in equations (11) and (12).

If $V_i \leq V_{av}$ **then** $Q_i = 0$ .... **(11)**

If $V_i > V_{av}$ then $Q_i = 1$ .... **(12)**

Where "I" is the index of each pixel in the block, V_{av} is the average value of all the pixel values and V_i the value of all the pixels in the given block. As per the ADRC, a 3 x 3 encoded block is shown in Figure 8 as an example.

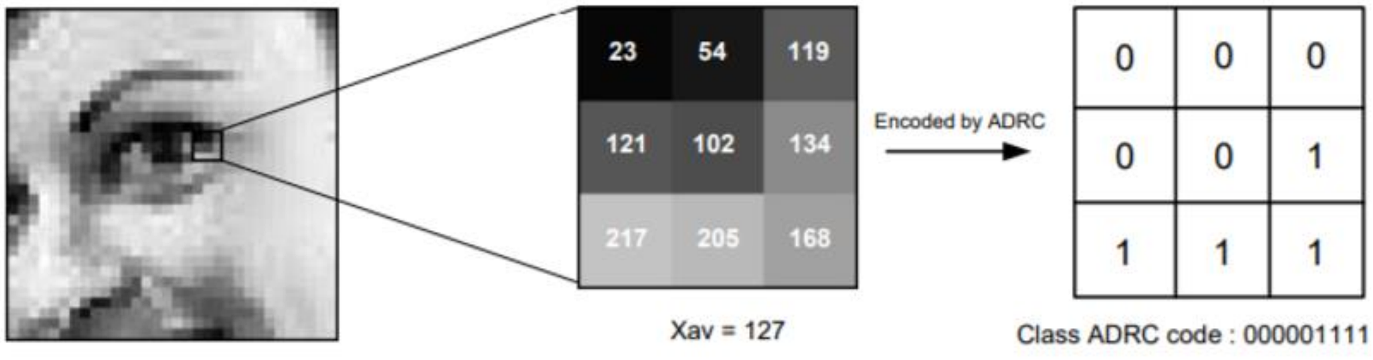


Figure 8 The ADRC Code of a 3 x 3 Block

II. ***Classification on Basis of local Complexity***

The ADRC will be implemented for successful classification based on the structure of images but like in every case, some structure noise or artifacts (that can arise from the code) can have the same ADRC pattern. So, other classification methods are proposed to be combined with the ADRC which on its own is unfortunately not enough. These classifications methods are generally

1) The Local Entropy Approach (BOVIK, 2000)
2) The Dynamic Range (DR) (PRATT, 2007)
3) The Standard Deviation (STD) (Shao, 2008)
4) The Mean Absolute Difference (MAG) (Ling Shao H. Z., 2008)

Most of the above mentioned are generally used to find the complexity of a local image.

H and Haan (PRATT, 2007), had proposed adding the Dynamic Range (DR) to the ADRC to solve the noise structure or the artifacts problem. The DR is working as a pre-set threshold factor to retrieve the contrast information. Thus, an extra bit of the DR was added to the ADRC to help solve the noise and artifact problem. The extra bit is defined in equations (13), (14), and (15).

If $(X_{max} - X_{min}) < T_r$ .... **(13)**

Then DR = 0, ... **(14)**

Otherwise DR = 1 .... **(15)**

Where X_{max} is defined as the maximum pixel value, whilst X_{min} is defined as the minimum value, T_r indicates a threshold value having "**r**" as the local region inside.

Unlike Hu and Han, Shao had designed a local entropy classification method that was designed based on information theory to further differentiate the complex and the flat areas.

By using the entropy (classification) equation, the local entropy H in the region R would be defined in equation (16) (PRATT, 2007),

$$\mathbf{H} = (-)\sum_{i=1}^{N} P_R(i)\, log_2\, P_R(i) \;\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \mathbf{(16)}$$

Where "i" is the aperture index, $P_R(i)$ is the probability of the pixels having specific values in the range of the aperture "i". N is the number of apertures. The other two methods of classification the MAG and the STD are used to find the complexity which is also proposed by Shao (Ling Shao H. Z., 2008) (Shao, 2008). Turns out the methods mentioned (STD and MAG) have an improvement in the classification results.

2.4) <u>Evaluation Method</u>

I. Image Quality Assessment

Some digital image operations such as reconstruction, treatment, compression, transmission, and acquisition can lead to various distortions that can severely affect the quality of the images. The final quality of the said images will reflect the performance of the used algorithms and/or equipment.

To optimize these systems and obtain a better performance, it is recommended to find a suitable measurement for the quality of both the images from the input and the images from the output. The image quality assessment can be generally divided into two parts the subjective and the objective method. In this report, the viewer will be a person hence the subjective method is more dependable than the counterpart (objective).

The only drawback is that the subjective method is time-consuming and hard sledding and it is also very difficult to be practically active because there will be different feedback from different viewers, and unfortunately Objective method cannot be implemented using MATLAB, it would require C language platform as one its requirements.

II. MSE and PSNR

MSE or the mean square error can help in estimating the quality of an image, the MSE score can only reveal the image quality to a certain extent. A point to be noted is that any two images processed in different ways can still get the same MSE value. According to the viewer's visibility, the one which has obtained a higher MSE score may perform better, based on people's vision solely.

The PSNR or the Peak signal-to-noise ratio is used in analog systems. the PSNR values are generally somewhere in between 30dB and 50dB, the latter is a better value. When the MSE score of two images is equal to zero, the PSNR of both those images will approach to infinite. The equations are present in Chapter 7.

3) Video Watermarking using Discrete Wavelet Transform

3.1) Discrete Wavelet Transform (DWT)

The Discrete Wavelet Transform is the "multi-resolution" description of a given image, where the decoding of a low-resolution image can be processed in sequences to a higher resolution (Xiang-Gen Xia, 1997). The DWT is based on small waves are known as wavelets, of limited duration, and constantly changing frequency. These wavelets of the DWT provide the spatial description and the frequency of an image (Hong, 2006). The DWT separates the signal into low and high-frequency parts.

The high-frequency part of the signal will have the edge component information. The low-frequency part of the signal will again be separated into a "high" and "low" frequency parts. The "high" frequency part is generally used for the watermarking method as the naked eye is less sensitive to minor changes in the edges (D. Kundur, 1998). The input image/frame is first decomposed by the DWT based on hierarchy into four frequency districts, namely high-high (HH), high-low (HL), low-high (LH), low-low (LL) as shown in Figure 9.

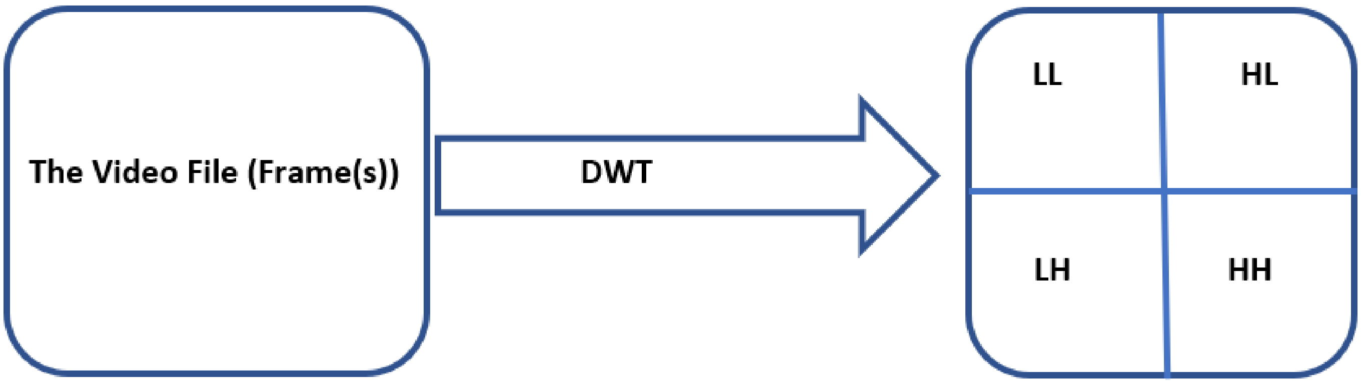


Figure 9 A Video Frame(s) DWT Frequency Districts

The DWT can decompose a two-dimensional image by more than one specific level. In the technique used in this paper, a 2 level DWT was used, and the HL (high-level) frequency district is used in the second level. Every frequency district is a matrix of the DWT coefficients at a given resolution. In Figure 10 the frequency distribution is produced by the second level (2-level) DWT decomposition.

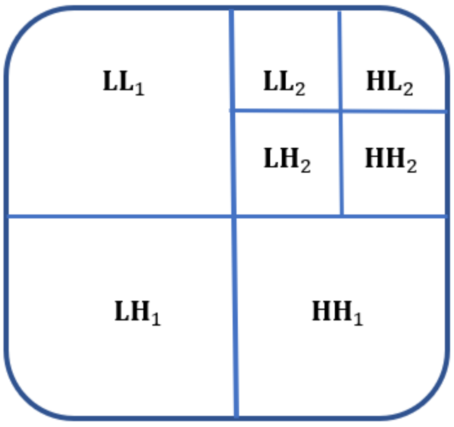


Figure 10 The Frame(s) DWT 2-level Frequency District

3.2) Schur Decomposition

The Schur triangulation (also known as Schur Decomposition) is a very helpful mathematical tool used in linear algebra metric analysis. The Schur Decomposition represents a major step in the SVD Decomposition (Charles F. Van Loan, 2012), it is said to be a very efficient tool for mathematics because it requires $\mathbf{n}^3$ flops for a n x n matrix. While the SVD generally needs about $11\mathbf{n}^3$ flops.

This can mean that the number of calculations performed in the Schur Decomposition is less when compared to the calculations performed in the SVD Decomposition (Mohammad, 2012). There are two versions of the Schur Decomposition, they are The Real Schur (B.Chandra Mohan, 2011) and the Complex Schur (B.Chandra Mohan, 2011). For a real-Square matrix A, the Real Schur Decomposition will result in two matrices, S and U as shown in equation (17).

Schur (A) = $(\mathbf{U} \times \mathbf{S} \times \mathbf{U}^{\mathbf{T}})$……………………………………………………………………………… (17)

In (1), S is a block in the Upper Triangular Matrix (the Real Schur form), and U is the Unitary Orthogonal Matrix. $\mathbf{U}^{\mathbf{T}}$ is the Conjugate Transpose of U. A small example will show the Schur Decomposition of a 3 x 3 matrix (A).

$$\mathbf{A} = \begin{bmatrix} \mathbf{160} & \mathbf{160} & \mathbf{159} \\ \mathbf{160} & \mathbf{159} & \mathbf{159} \\ \mathbf{161} & \mathbf{160} & \mathbf{158} \end{bmatrix}$$

After the application of The Schur Decomposition, the matrix (A) will split into S and U matrices.

$$\mathbf{S} = \begin{bmatrix} \mathbf{478.6664} & (-)\mathbf{0.7071} & (-)\mathbf{2.0418} \\ \mathbf{0} & (-)\mathbf{0.6664} & (-)\mathbf{0.5753} \\ \mathbf{0} & \mathbf{0} & (-)\mathbf{1.0000} \end{bmatrix},$$

&

$$\mathbf{U} = \begin{bmatrix} \mathbf{0.5778} & \mathbf{0.4077} & (-)\mathbf{0.7071} \\ \mathbf{0.5765} & (-)\mathbf{0.8171} & \mathbf{0.0000} \\ \mathbf{0.5778} & \mathbf{0.4077} & \mathbf{0.7071} \end{bmatrix},$$

The S matrix is triangular, and it's also quite similar to matrix A, as per the definition of the matrices' similarities (Charles F. Van Loan, 2012). The matrix S also has the same set (multiple sets) of the eigenvalues in A. the eigenvalues set(s) are contained within the diagonal of matrix S, because as mentioned previously S is a triangular matrix.

Because the eigenvalues in Schur Decomposition are very stable (B.Chandra Mohan, 2011), the diagonal in matrix S was the preferred choice for the watermark embedding.

Chapter 6: Design Ideas and Concept Selection

1) Detection and Tracking of a Moving Object

1.1) Background Subtraction (used)

This approach will compute a model of the background of the images (frames) and keep updating the model. The algorithm utilized has been taken from (Liyuan Li, 2003), this algorithm is more robust and has a very good performance, though it possesses higher calculation requirements than the other algorithms in (Piccardi, 2004) & (C. Stauffer, 1999). However, the performance obtained at this stage will greatly affect the upcoming stages as well, which is why a highly accurate and robust algorithm is needed.

The background of a video or image(s) is those objects which stay in one place (roads, buildings, walls) or those objects that move in small amounts (trees swaying, grass swaying) within the scene. The background subtraction model will first build a histogram-based statistical model of the video/image background for every pixel.

This model will then collect the color cooccurrences in the given video and then to decrease the computational load applies quantization. This selection of the color cooccurrences will help in increasing the accuracy of the algorithm within the presence of the slightly moving background objects. Then the algorithm will apply the “Bayes decision” rule for classifying the background and the foreground. This “Bayes decision” rule expediates the use of posterior probabilities of the feature vectors and will compute the very probability of a feature vector (the color cooccurrences) that belong to a specific class.

The classification will then be made easily by comparing the probabilities. The important property of any given effective background subtraction algorithm is to adapt the background model to the varying conditions in the video/image. Some of these conditions include (but are not limited to) changes in illumination (passing clouds that block the sun certain times during the day), changes in time throughout the day, etc.

with these conditions that may,/may not be visible, the background subtraction algorithm that is implemented will update the respective background model at each frame as follows,

After the classification is done the new features from the background of the video/images are collected and utilized using a weighted average filter (infinite pulse IIR filter) which will update the model, meaning that both the old and the new features will have an effect on the background model but with different degrees. This when combined with the “Bayes decision” rule, both the slow and the quick changes in the video/image(s) can be accounted for and these can be used to update the statistics. Lastly, the algorithm maintains a “reference image” of the background.

The basic block diagram is shown in Figure 11, Due to the similar pixels present in the foreground and the background, many holes can be formed in the binary mask of the moving object(s). to solve this problem (called the hollow phenomenon), erosion and/or dilation of the morphological operations (Rafael C. Gonzalez, 2008) is used (explained in detail in 1.4).

It will in addition to filling the hollow also remove most of the noise of the binary image. It can therefore provide a full mask of that specific moving object(s) to obtain a better extraction later. An example of the background subtraction module can be seen in Figure 2 and Figure 3.

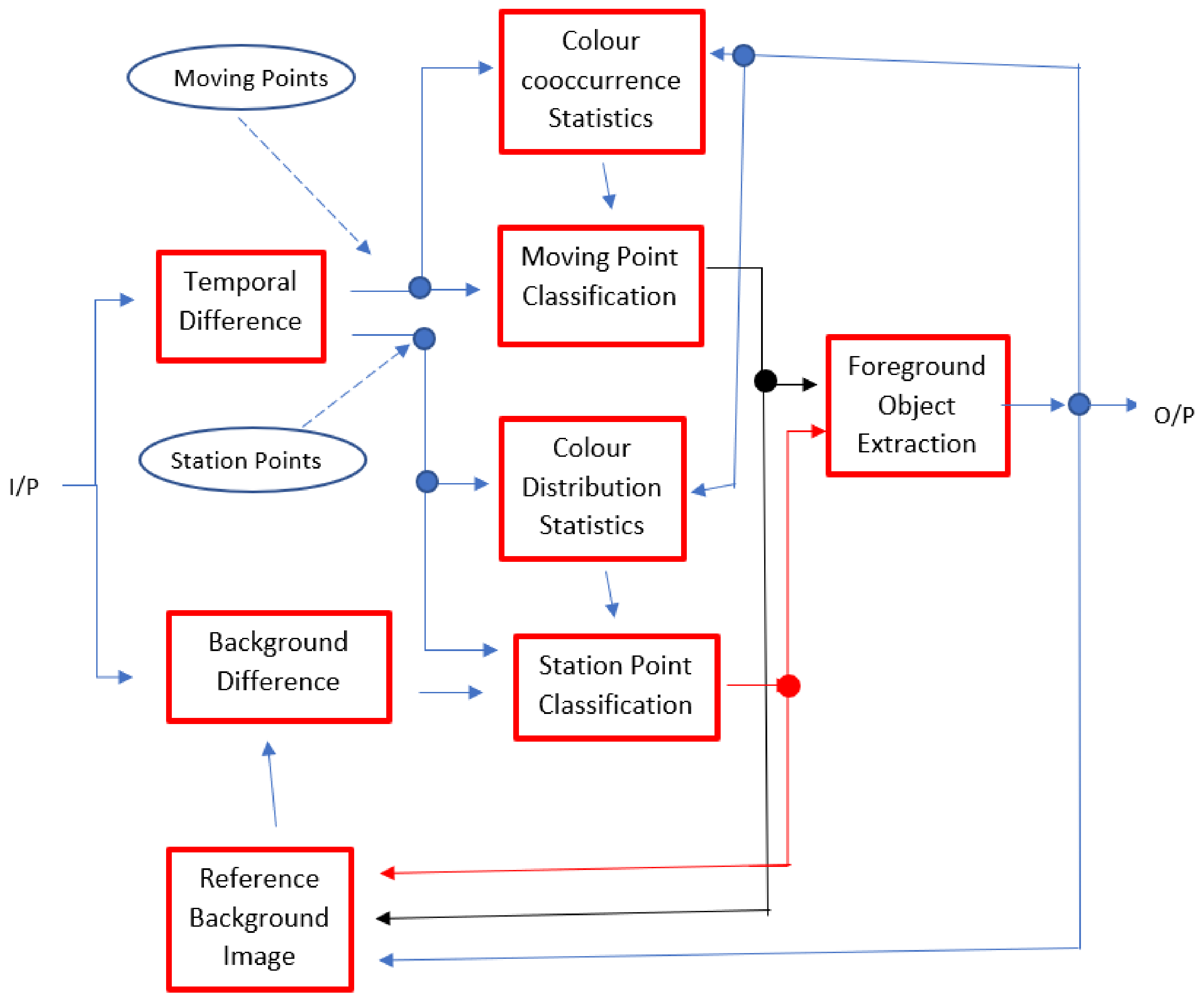

Moving Points
Colour cooccurrence Statistics
Temporal Difference
Moving Point Classification
Station Points
I/P
Colour Distribution Statistics
Foreground Object Extraction
O/P
Background Difference
Station Point Classification
Reference Background Image

Figure 11 Block Diagram of Background Subtraction Algorithm

1.2) <u>Blob Detection (not used)</u>

In the blob detection model, the foreground pixels (which can be obtained from the background subtraction model) are put together by using a Contour Detection Algorithm (Satoshi Suzuki, 1985) in the current frame. The algorithm (Contour Detection Algorithm) puts together the individual pixels into disconnected classes and will start finding the contours surrounding every class.

Every class is marked as a Candidate Blob (CB). This Candidate Blob (s) will then be checked based on their sizes and the small Candidate Blob(s) will be removed from the algorithm to minimize any false detections. An example of the blob detection is shown in Figure 12.

Even though in the MATLAB software this algorithm was quite useful, by simple eye judgment it was seen that the Morphological Image Processing (explained in 1.4) had done a better task of removing noise and identifying the object as well as being far simpler for the hardware to process it.

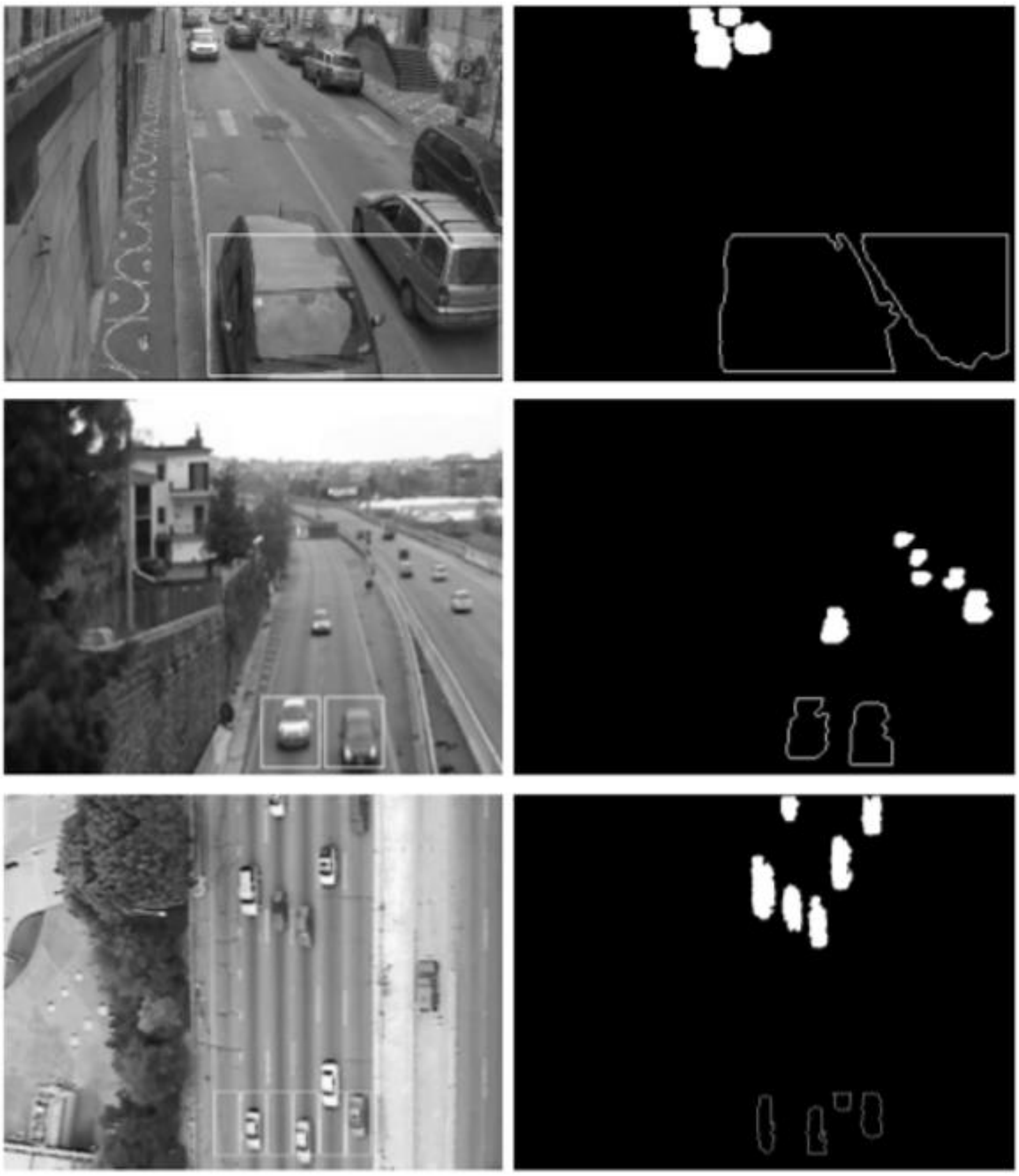


Figure 12 (Left) The Original Video Frame, (right) the result of Blob Detection Module (Salvi, 2012)

1.3) <u>Blob Tracking (not used)</u>

The proposed method will help to track each blob within the successive frame(s) of the video. Just after blob-analysis, the blobs along with their respective bounding boxes and centroids will be removed from each frame. The two objects that are closest to the adjacent frames will be connected. Euclidean distance will be used to measure the distance between the respective centroids.

The area of the object passing will also be taken into consideration to help enhance the object tracking method. For every moving object in the current frame, the object(s) with minimum size and/or distance will have to be searched in the previous frames. The basic match functions are shown in equations (18) and (19).

$$Distance^n = \sqrt{(\boldsymbol{x}\,\boldsymbol{B}_c - x\,\boldsymbol{B}_n)^2 + (\boldsymbol{y}\,\boldsymbol{B}_c - y\,\boldsymbol{B}_n)^2} < \delta \ldots\ldots \textbf{(18)}$$

$$\rho^n = |(\,w\boldsymbol{B}_c * \boldsymbol{h}\,\boldsymbol{B}_n) - (\,w\boldsymbol{B}_c * \boldsymbol{h}\,\boldsymbol{B}_n)| < \gamma \ldots\ldots \textbf{(19)}$$

Where n is the previous frame number. $\boldsymbol{B}_c and\ \boldsymbol{B}_n$ are the blob in current and the blob in the previous frame respectively. $\gamma\ and\ \delta$ are the thresholds.

If the $Distance^n$ is minimum and the criteria $Distance^n < \delta$ and $\rho^n < \gamma$ are met, then the object(s) in the current frame will be the very object in the previous frame and the blob will identify that object and assign it a label. This method was a concept that was fiddled around solely to count the number of objects passing through a given point. This report chose not to go ahead with this concept as it was believed to over complicate the work done so far, as the first part of this report was only to detect and track the moving object(s).

1.4) <u>Morphological Image Processing (used)</u>

The morphological image processing is a collection of non-linear operations that are related to the morphology (shape) features in a given frame (image). As per (Owens, 1997), the morphological operations will depend only on the relative pixel value ordering, and not on their numerical values. They are therefore suited for the processing of binary images. These operations can be applied to the grayscale image formats so that their light transfer functions (Efford, Digital Image Processing: A Practical Introduction Using Java, 2000) are not known or they are of no interest or are of minor interest.

The morphological methods review an image (frame) with a small template that is called a structuring element. This structuring element is positioned at all possible locations within the image (frame) and it is compared with the neighborhood of nearby pixels. Some of the operations will see whether the said element will “fit” in the neighborhood or whether the element will hit or intersect the neighborhood.

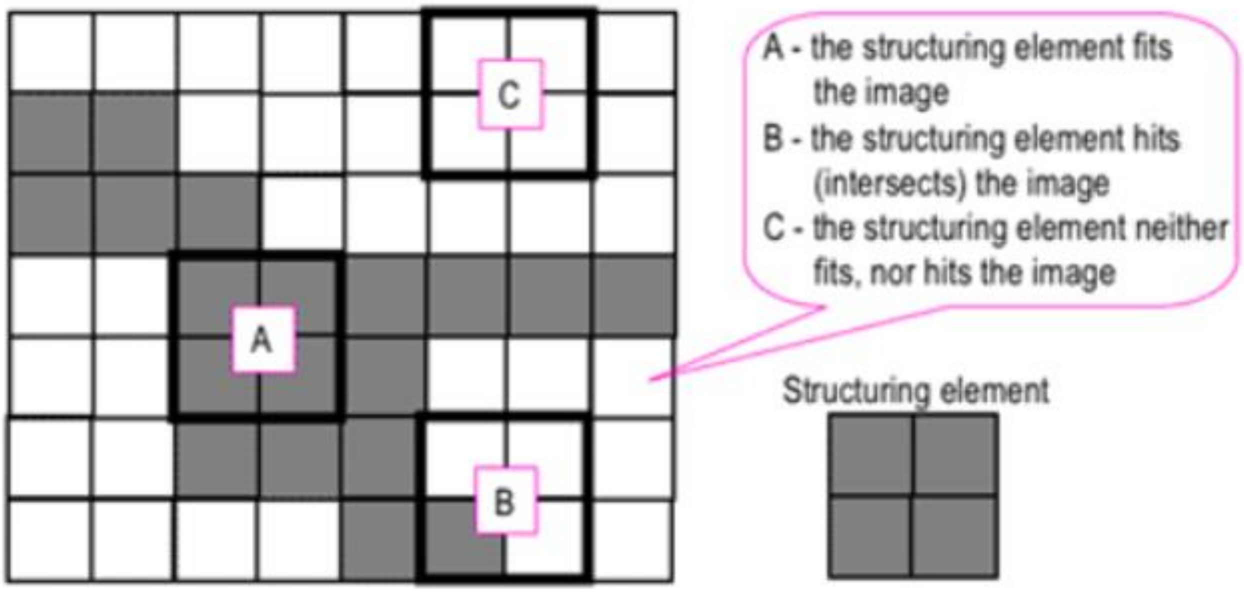

C
A
B
A - the structuring element fits the image
B - the structuring element hits (intersects) the image
C - the structuring element neither fits, nor hits the image
Structuring element

Figure 13 An Image probed with a structuring Element (Efford, 2000) here the White Pixels have Zero and the Grey Pixels have non-zero values

A binary image undergoing morphological operation(s) will create a new binary image where the pixels have a non-zero value if the test turns out to be successful at that location in the input image.

- The structuring element is a small binary image, meaning it is a small matrix of the pixels each having a binary value.
- The dimensions of the matrix will specify the structuring element size.
- The shape of the structuring element is determined by the pattern of ones and zeros.
- One of the structuring element pixels is generally the origin, but the origin can also be outside the structuring element

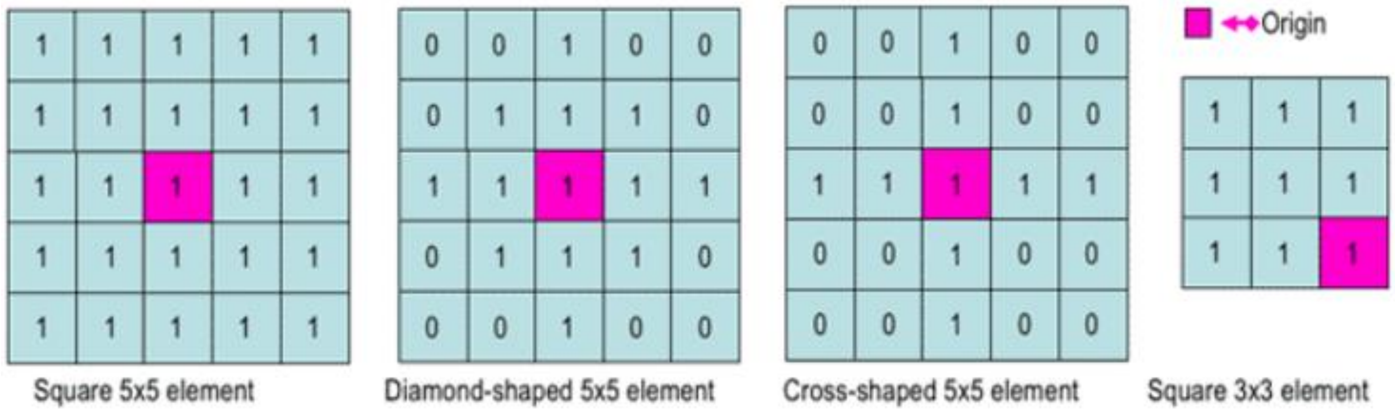


Figure 14 Simple structuring Elements examples (Efford, 2000)

I. <u>***Erosion***</u>

for a binary image "f" by a structuring element "s" (*f* ⊖ *s)* the erosion operation will produce a new binary image g = (*f* ⊖ *s)* with ones in every location (x,y) of the structuring element's origin, where the structuring element "s" will fit the input image f,

meaning that g(x,y) = 1 if s fits f,

otherwise 0

for every pixel coordinates (x,y).

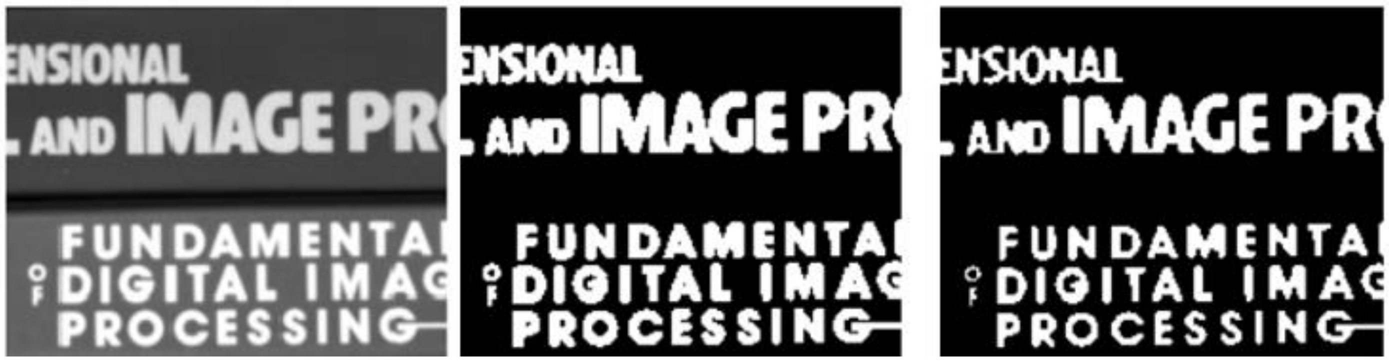


Figure 15 (left) The Grayscale Image, (middle) the Binary Image by Thresholding, (right) Erosion of an Image (Efford, 2000)

The erosion of a small square structuring element (such as a 2x2 to 5x5) will shrink an image by removing a layer of pixels both from the outer boundaries and the inner boundaries of the region. The gaps and holes between the different regions will become larger and the small details will be eliminated.

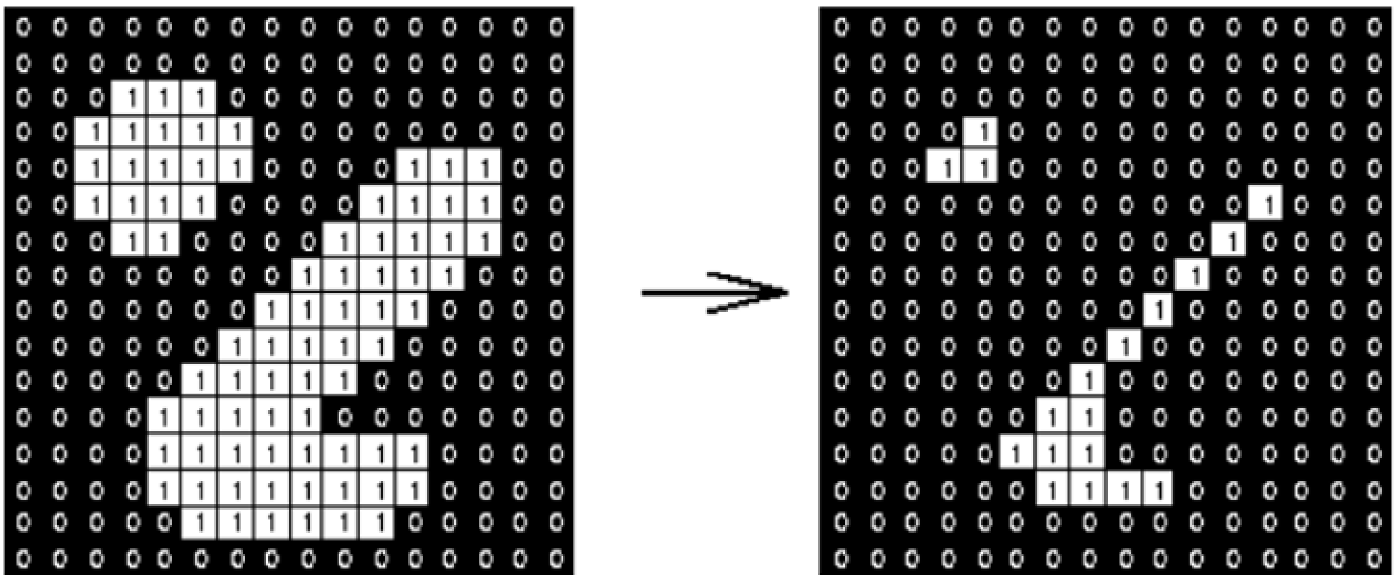


Figure 16 The Erosion of a 3x3 Square structuring Element (Efford, 2000)

The comparatively larger structuring elements will have a more pronounced effect, the result of the erosion with a larger structuring element will have a more pronounced effect. If S_1 and S_2 are two structuring elements that are identical in shape with S_2 having twice the size of S_1, then equation (20) occurs where,

$$\boldsymbol{f \ominus S_2 \approx (f \ominus S_1) \ominus S_1} \dots\dots\dots\dots\dots\dots \mathbf{(20)}$$

Erosion will remove some small-scale details from the binary image(s), but it will simultaneously reduce the size of the regions that are of interest as well. By subtracting the eroded image(s) with the original image(s), the boundaries of each region can be obtained which is shown in equation (21),

$$\boldsymbol{b = f - (f \ominus s\,)} \dots\dots\dots\dots\dots\dots \mathbf{(21)}$$

where f is the image containing the regions, s is a 3x3 Structuring Element, and b is the image containing the region boundaries.

II. <u>Dilation</u>

for a binary image “f” by a structuring element “s” (*f* ⊕ *s)* the erosion operation will produce a new binary image g = (*f* ⊕ *s),* with ones in every location (x,y) of the structuring element’s origin, where the structuring element “s” will fit the input image f,

meaning that g(x,y) = 1 if s fits f,

otherwise 0

for every pixel coordinates (x,y). when compared to Erosion Dilation has the opposite effect, Dilation will add a layer of pixels to the inner and the outer boundaries of the regions,

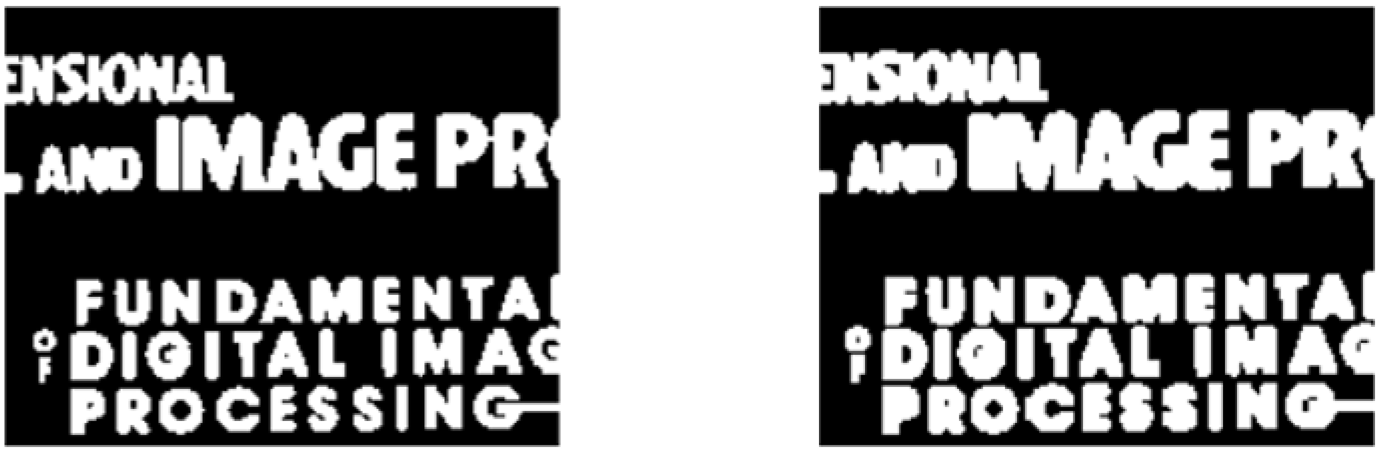


Figure 17 (left) A Binary Image, (right) Dilation of a 2x2 Square structuring Element (Efford, 2000)

The gaps between different regions and the holes enclosed by a single region will become smaller and the small intrusions into the boundaries of a region will be filled.

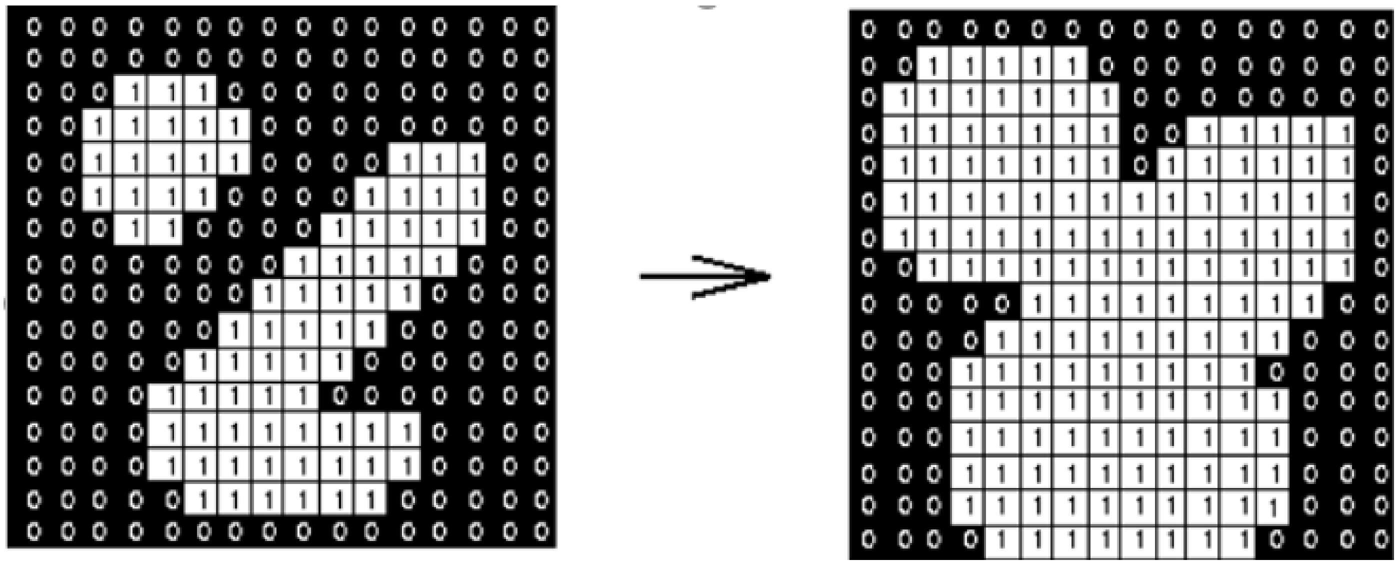


Figure 18 Dilation of a 3x3 Square structuring Element (Efford, 2000)

The results of either dilation or erosion will be influenced by the shape and the size of the structuring element. They both are dual operations, meaning that they have opposite effects. Let f^c be the complement of an image f, meaning the image produced by replacing the 0 with 1 and vice-versa. Formally written as equation (22),

$$f \oplus s = f^c \ominus s_{rot} \quad \text{.......} \quad (22)$$

where S_{rot} is the structuring element s rotated by 180^0. If a structural element is symmetrical w.r.t rotation, then S_{rot} will not differ from s. If an image (binary) is considered to be a collection of connected regions of pixels set to 1 on the background where the pixels set to 0, then case erosion is the “fitting” of a structuring element to these regions and dilation will be the fitting of a structural element into the background followed by the result inversion.

This combination of the background subtraction with the morphological image processing has shown to have the least amount of burden on the hardware and also most importantly have the least amount of noise. By basic trail, an error this specific combination seemingly is much better than Blob Detection, Blob Tracking, Ransac, Optical Flow Algorithm, and/or KLT.

2) Video Quality Enhancement

2.1) Simulation Frame (not selected)

A slightly different method would involve repairing a low-quality video enhancement algorithm, here a simulation of the “offline” process as well as the filtering (run-time) process. A step by step methodology that is generally common for repairing any low-quality video enhancement algorithm is shown in Figure 19.

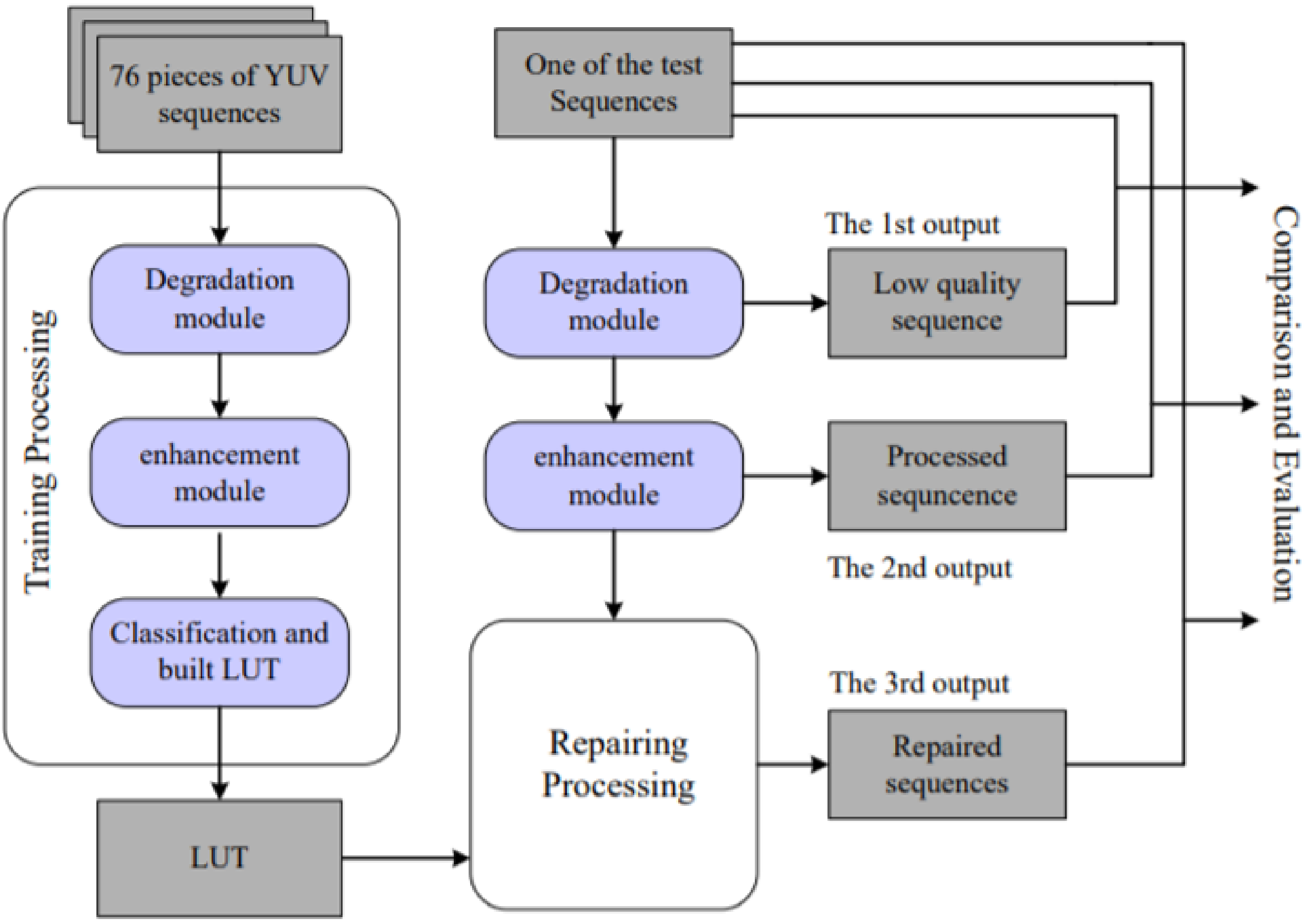


Figure 19 The Common Procedure of the Three Embodiments

For any given experiment the video processing modules shown in Figure 19 will be computed by the C language.

a) degradation of the respective test sequences will be treated as “lower-quality” videos that need to be enhanced.
b) For evaluation purposes, to enable an MSE and SSIM (Umme Sara, 2019) calculation, the original test sequences will be compared to the resulting test sequences for reference(s).
c) A plethora of YUV (a specific video format) must be trained to obtain optimized coefficients for each training process.
d) The degradation module as well as the enhancement module will be the same In the training process as the ones used in the test sequence.

This algorithm seems to show promise and also give really good results, the sole reason that this method was not considered is that this algorithm “cannot” be implemented in MATLAB it can only “be done” using C, which was not the chief aim of this report (to perform a task using MATLAB).

2.2) <u>Processing Module Selecting</u>

I. *<u>Coding artefacts Restoration Algorithm (not selected)</u>*

In the Spatial domain, the linear filters such as an averaging filter (Berge, 1993) and the non-linear filters include a median filter. The average filters are based on the concept to re-assign every pixel's location with the average value that is obtained from the pixel's neighbors in a pre-set filtered aperture. This way the sharp transitions in intensity will be reduced. Figure 20 is the result of the images after using a median filter to minimize the coding artifacts.

In the Frequency domain the sharp transitions in the intensity which arise from (noise, coding artifacts, and/or blocks are generally present in the form of high-frequency Fourier transform signals. This is why a low-pass filter would be the ideal choice to solve this "noise" problem. Some of the filters are Butterworth (P. Thévenaz, 2000), Gaussian, and Idea.

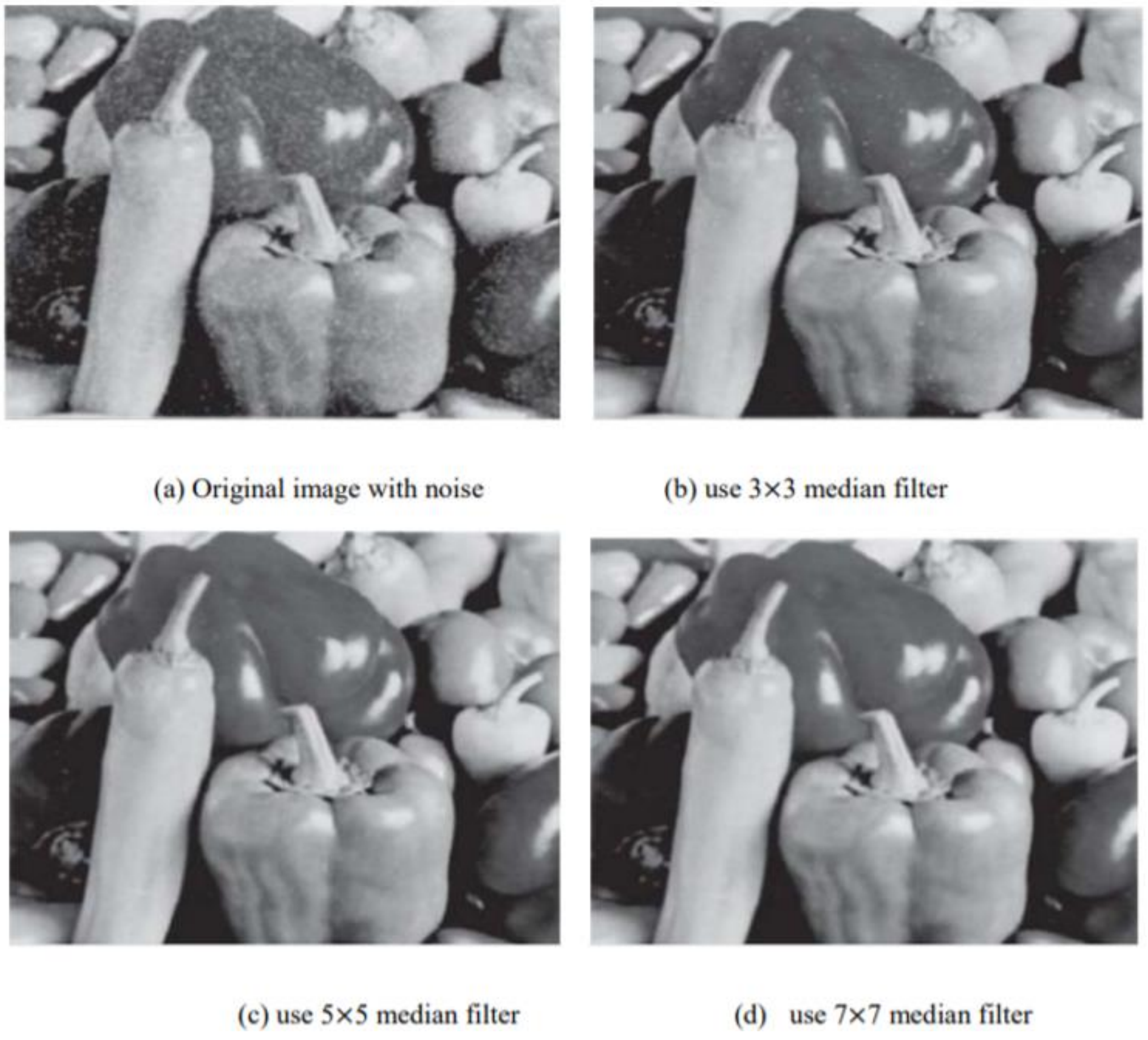

(a) Original image with noise
(b) use 3×3 median filter
(c) use 5×5 median filter
(d) use 7×7 median filter

Figure 20 Noise Reduction with the help of the Median Filter (C.B. Atkins, 7-10 Oct. 2001)

To repair the algorithm, the compression module will be simulated as a degradation process using the C language. This is closely based on the JPEG compression standard. The JPEG quality will be set to 20 so that sufficient coding artifacts are ensured for a sharp contrast with the other resulting outputs. For example, purpose a screen-shot of the "bord" YUV sequence post compression is shown in Figure 21. The result obtained may not be proper as generally, this method requires an edge adaptive method in addition to get better results. Like in the Simulation frame case this method also "cannot" be implemented in MATLAB and requires C for its use. Which is why this method was not selected.

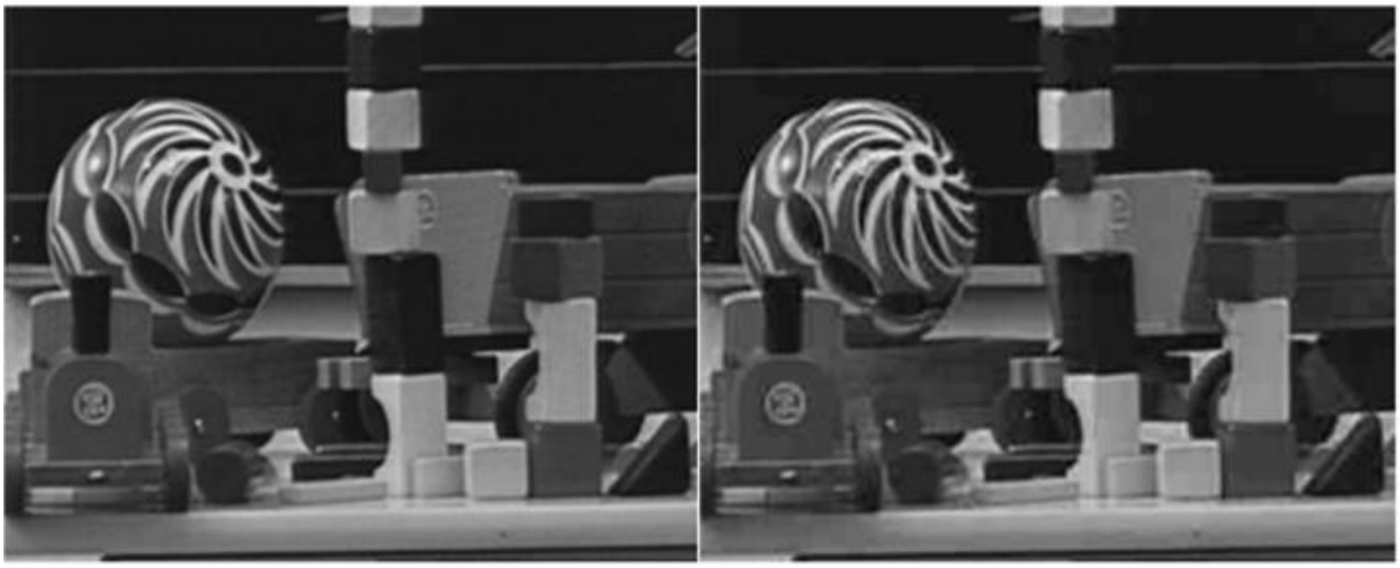


Figure 21 The Results originated from Compression (C.B. Atkins, 7-10 Oct. 2001)

II. ***Deblurring Algorithm (selected)***

A proposed peaking filter program is utilized in simulating the video/image(s) enhancement for the de-blurring procedure. In this proposed case the alpha is set to be 0.2, and the other three coefficients are set as (alpha, 1+2 alpha x alpha) for the three adjacent pixels values. The peaking process will then be embedded with the given coefficients (-0.2, 1.4, -0.2) in the vertical direction as well as the horizontal directions. The effect of this program is shown in Figure 22. This method was selected because it "could" be used in both MATLAB and C Programming Language.

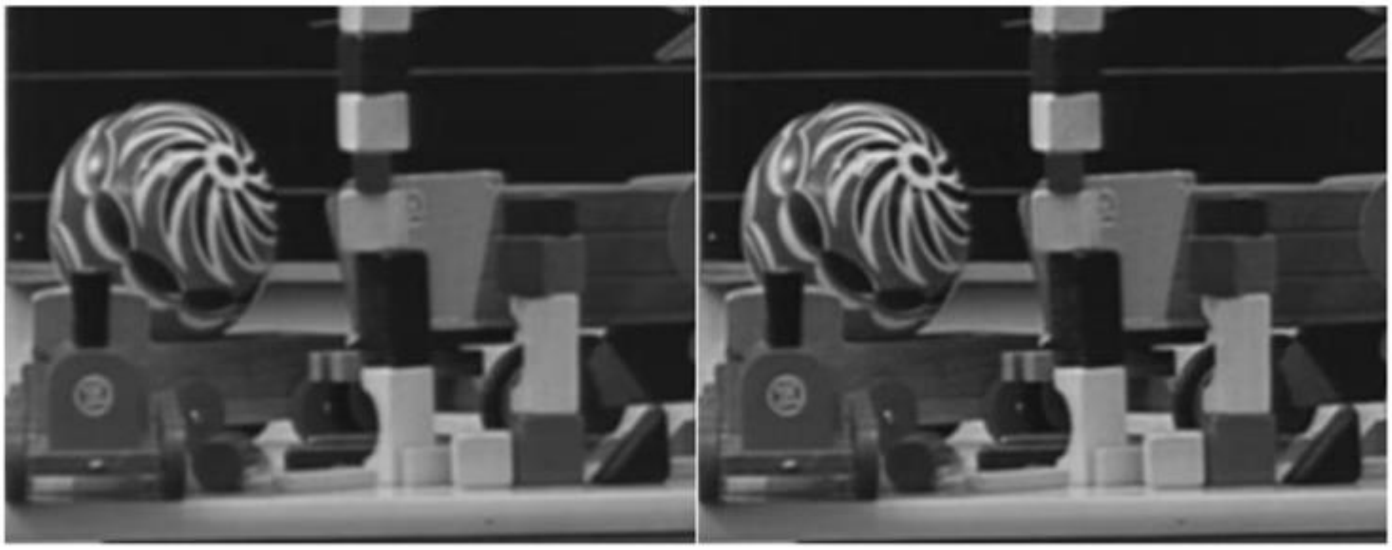


Figure 22 A Blurred sequence before and After the peaking filter (C.B. Atkins, 7-10 Oct. 2001)

III. <u>Resolution upscaling Algorithm (not selected)</u>

The Resolution upscaling algorithm also known as the resolution conversion simply put is getting a higher resolution image from a lower resolution image. In the Spatial Domain, the basic method for this is to increase the pixel number by image interpolation (P. Thévenaz, 2000). The most general or rather the most preferred interpolation technique (H. Hu, 10-14 Jan. 2007) is the linear resolution upscaling algorithm. Figure 23 will show the result obtained after using the resolution upscaling using interpolation.

In the Frequency Domain, the resolution upscaling is quite similar to the sharpness enhancement obtained increasing the pixel locations. To hold the high-frequency content a high pass filter is used to help get a better output. Recently an advanced non-linear resolution upscaling algorithm was developed it was the Kondo's data dependant interpolation filter (Ling Shao H. H., 16 Sept.-19 Oct. 2007), and the CBA (L.Shao, January 2007).

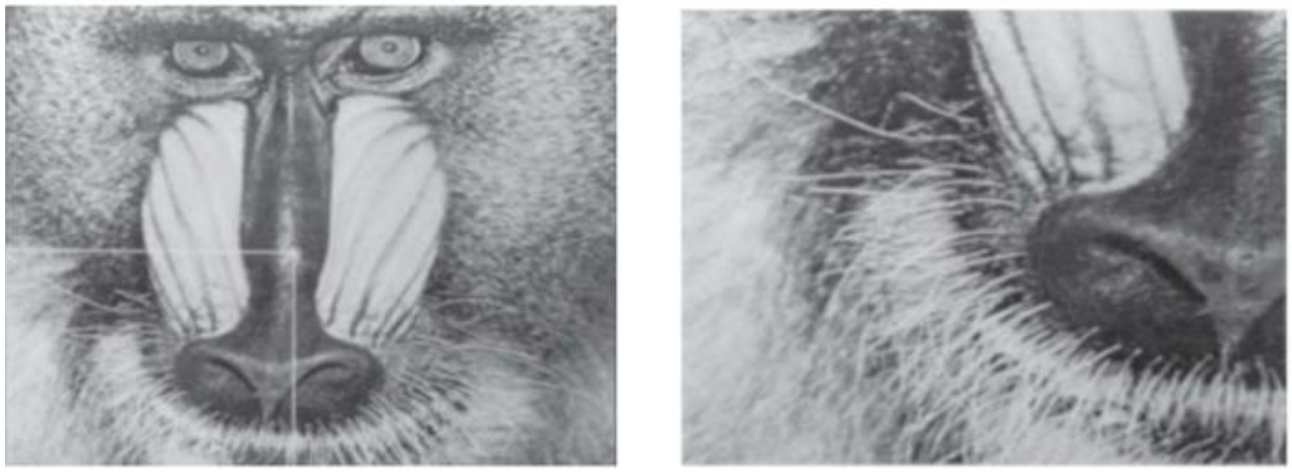


Figure 23 (Right) the Original Image, (left) the Interpolation for a 2:1 Magnification (C.B. Atkins, 7-10 Oct. 2001)

To repair this algorithm, a downsampling process is simulated by C programming to obtain a low-resolution sequence from the original sequence. In this program the original was cut into 4-pixel blocks, then the pixel value is then the average of the corresponding block in the test sequence(s).

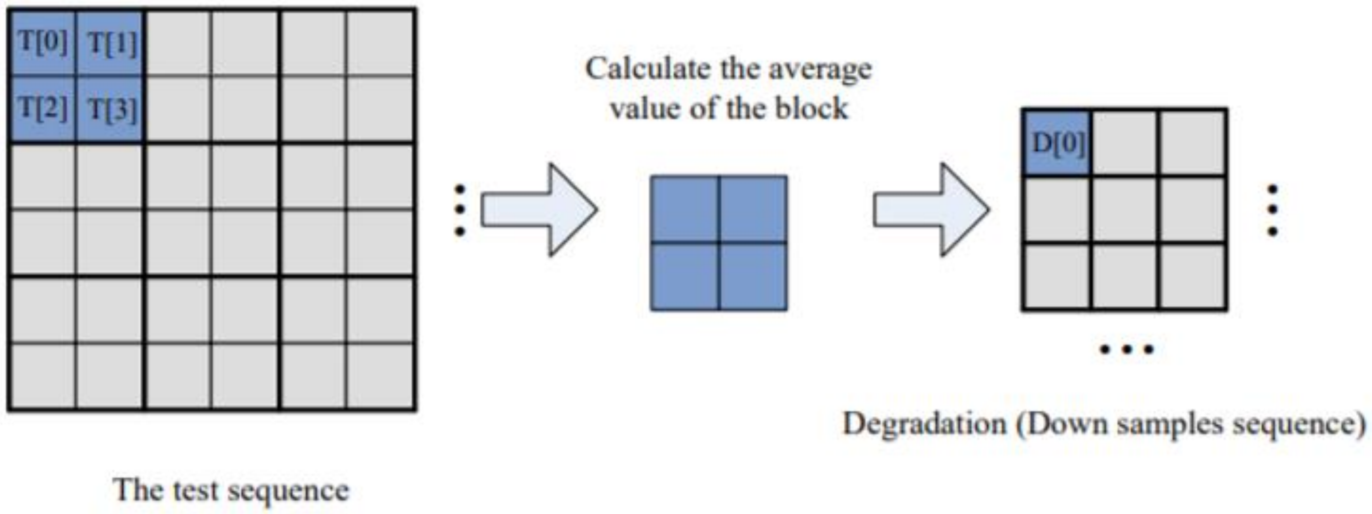

T[0]
T[1]
T[2]
T[3]
The test sequence
Calculate the average value of the block
D[0]
Degradation (Down samples sequence)

Figure 24 The Down Sampling Process

The graphic in Figure 24 is showing the correspondence between the test sequence and the degradation (the low – quality video input). The blocks T[0] to T[3] is the 4-pixel block within the test sequence, whereas D[0] is the sole pixel in the degradation.

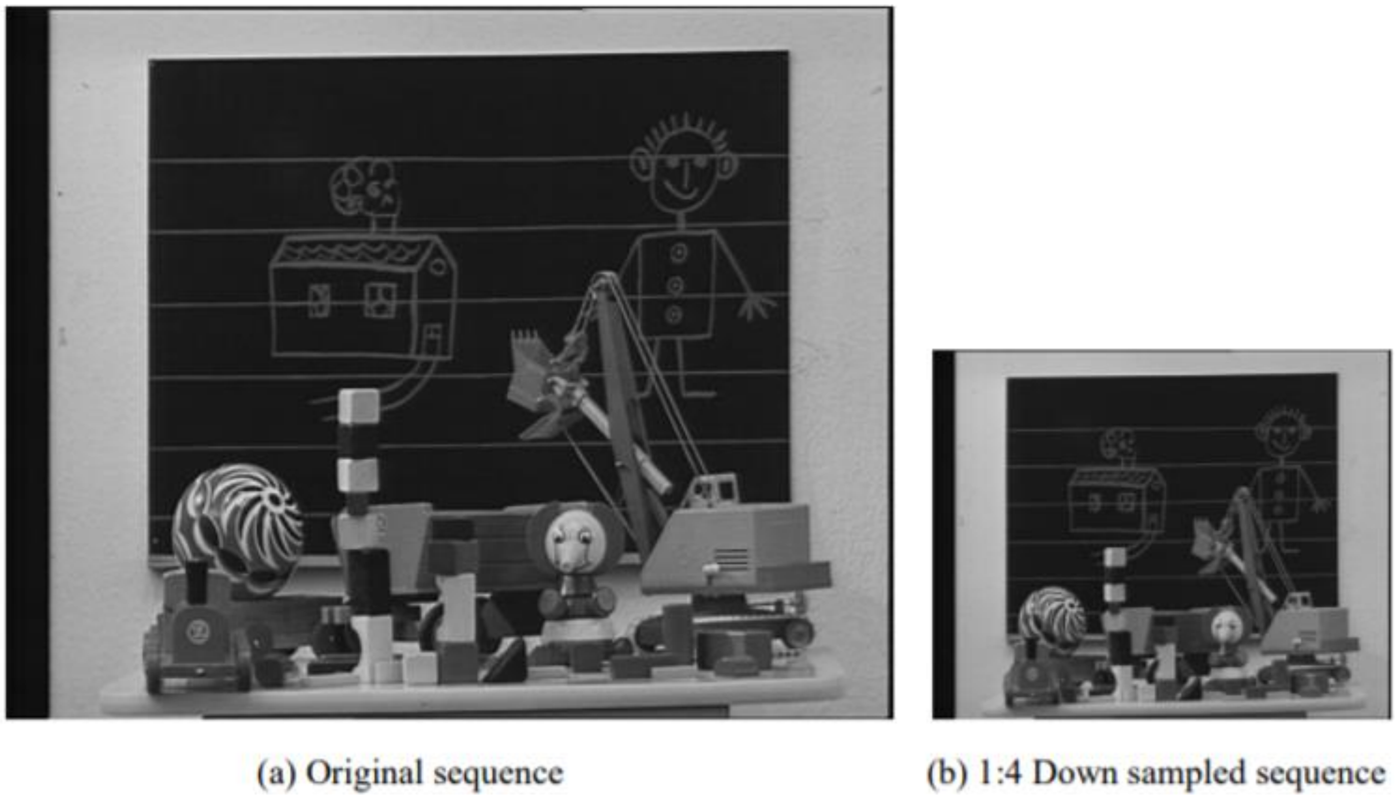


Figure 25 The Down Sampling Process (C.B. Atkins, 7-10 Oct. 2001)

A Bilinear processing program for the images is simulated to become the resolution upscaling program. But currently, the low-resolution video sequence is the input, which also will be cut into a 4-pixel block(s). this new 4-pixel block(s) can be manually calculated using the equations (23), (24), (25), and (26):

B[0] = (0.75 X D[0] + 0.25 X D[1]) X 0.75 + (0.75 X D[2] + 0.25 X D[3]) X 0.25 …………….... (23)

B[1] = (0.25 X D[0] + 0.75 X D[1]) X 0.75 + (0.25 X D[2] + 0.75 X D[3]) X 0.25 …………….... (24)

B[2] = (0.75 X D[0] + 0.25 X D[1]) X 0.25 + (0.75 X D[2] + 0.25 X D[3]) X 0.75 …………….... (25)

B[3] = (0.25 X D[0] + 0.75 X D[1]) X 0.25 + (0.25 X D[2] + 0.75 X D[3]) X 0.75 …………….... (26)

Over here, the blocks D[0] to D[3] are the pixel blocks present in the input down-sampled sequence, the blocks B[0] to B[3] is the pixel block present in the desired sequence. The relation between these two sequences is shown in Figure 26.

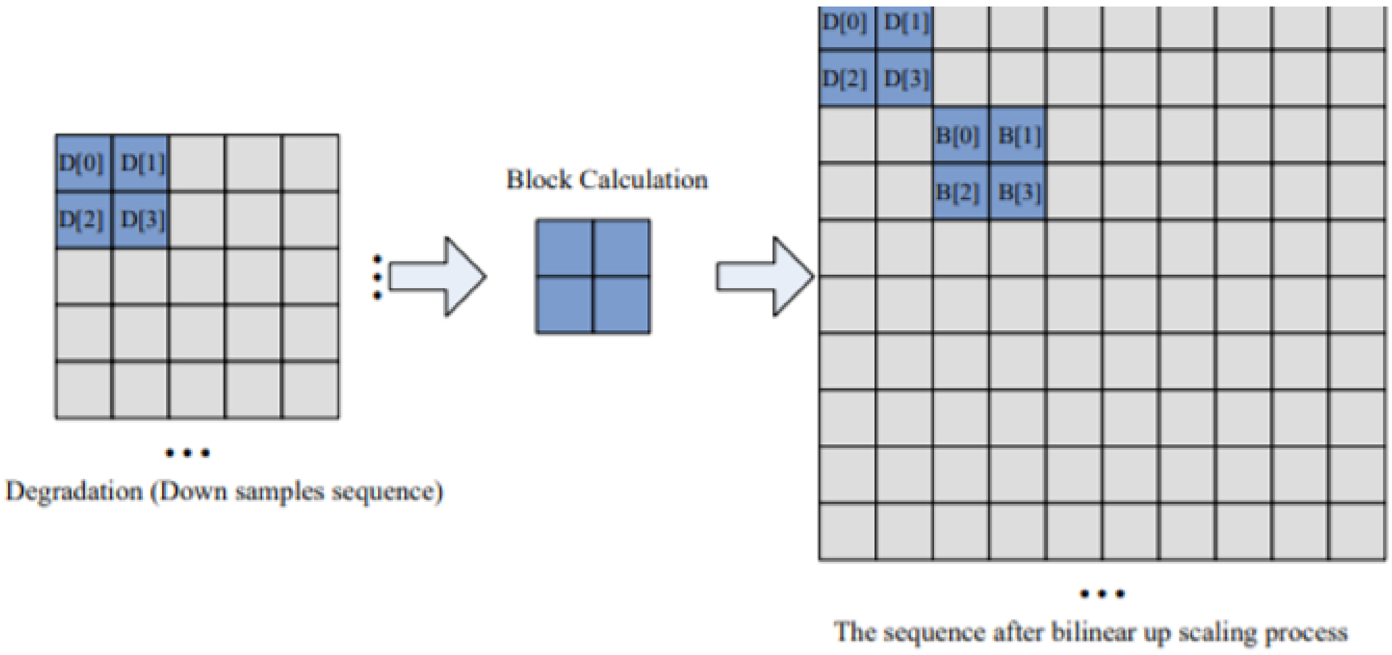

D[0]
D[1]
D[2]
D[3]
Degradation (Down samples sequence)
Block Calculation
B[0]
B[1]
B[2]
B[3]
The sequence after bilinear up scaling process

Figure 26 The Down Sampling Process

The resulting output is shown in Figure 27,

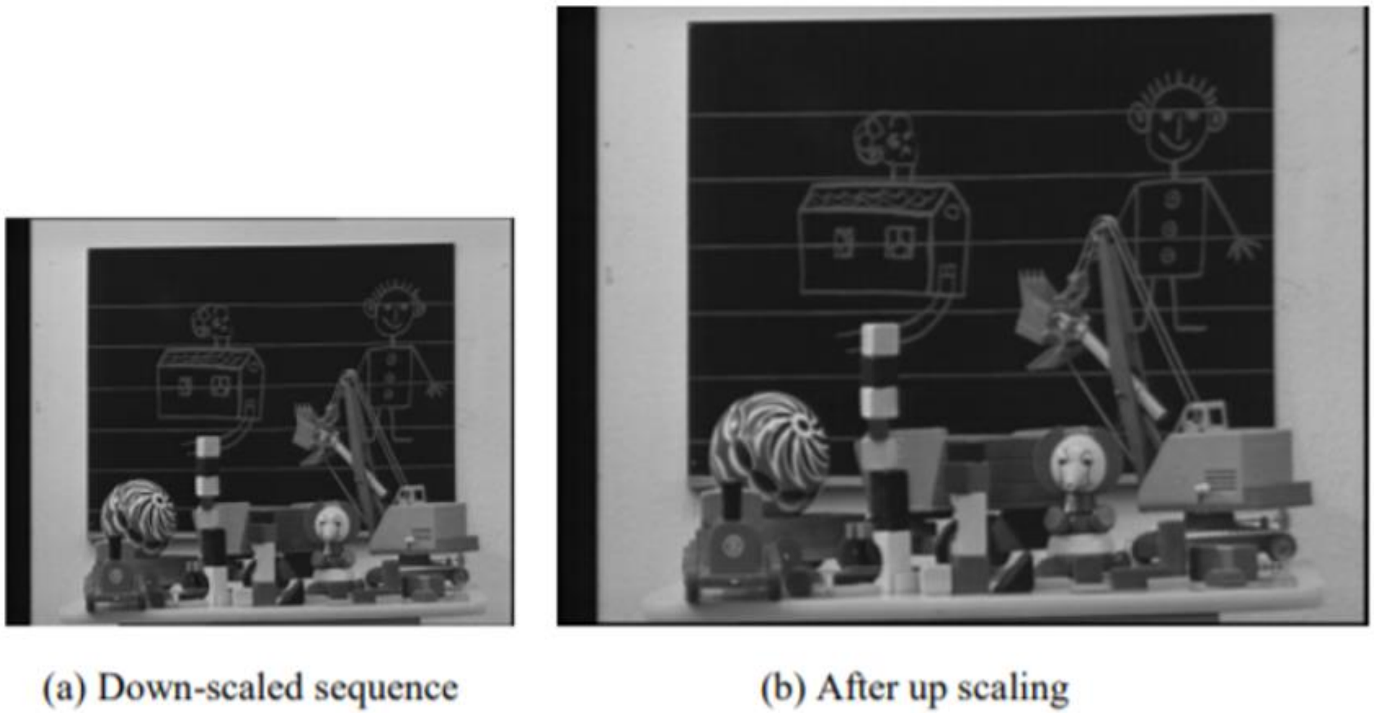


Figure 27 The Bilinear upscaling Process (C.B. Atkins, 7-10 Oct. 2001)

This concept was fiddled around to try and make sure the user can see manually by changing the resolution of the range of sequences to see how the deblurring algorithm had worked, it would make a great evaluation algorithm to test the deblurring algorithm by showing the magnification version of the image(s) to be manually judged. Before this algorithm could be studied further it was again more properly used in C and unfortunately, MATLAB perhaps due to a glitch or bug cannot handle this algorithm throughout the entire video.

3) Video Watermarking using Discrete Wavelet Transform

3.1) Video Frames and Video Watermarking

The host is a digital watermarking method is generally the location where the watermark is embedded. The host can be either audio, a video, or an image. A video sequence consists of a large number of frames, where each frame is considered to be an image. So, the image watermarking method can also be used (with some adjustments to the code) for video watermarking.

3.2) The Proposed DWT-SCHUR Method (selected)

This report uses a blind robust digital video watermarking method, the proposed method will apply both the Discrete Wavelet Transform (DWT) and the Schur transform on the video where a binary watermark will be embedded and then distributed within the video frames.

The proposed DWT-Schur watermarking method consists of 2 procedures. The first procedure will embed the watermark file in the video clip. The other procedure will extract it from the watermarked video obtained after the first procedure. The two procedures are explained below,

I. Watermark Embedding Procedure

The proposed DWT – SCHUR watermarking algorithm starts by embedding the watermark into the video frame(s). the detailed embedding procedure is,

1) First the input video clip V is selected
2) The video clip V is divided into video clip scenes Vs
3) Each frame from the video scene is then processed using DWT and SCHUR (which will be described in steps 4 to 11)
4) Every video frame F is converted from RGB to YUV color format

   (Application of Level-2 DWT)

5) The algorithm will calculate the level 2 DWT for the Y (luminance) matrix in every frame F. this procedure will generate an additional seven DWT sub-bands namely the $\mathbf{LL_1, LL_2, HL_2, LH_2, HH_2, LH_1, HH_1}$,

   (Application of SCHUR)

6) The Schur operator is applied to the $\mathrm{HL_2}$ sub-band. The Schur operator will then decompose the sub-bands coefficient matrix into two. As shown in equation (27),

$$\mathbf{HL_2 = U_{HL_2}\, S_{HL_2}} \quad \text{(27)}$$

   (Embedding Procedure)

7) The watermark image is rescaled so the size of the watermark will be equal to the size of the $\mathrm{HL_2}$ sub-band, which will then be utilized for embedding.
8) The binary bits of the watermark $\mathrm{W_{V_{si}}}$ will be embedded into the diagonal matrix $\mathrm{S_{HL_2}}$ by substituting the watermark bit(s) W_i with the eighth Least Significant Bit [equation (28)].

$$\textbf{LSB}\ (S_{HL_2}\ (i,i)) = W_{V_{si}}\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (28)$$

(The Video Reconstruction)

9) the inverse Schur operation is applied on the modified S'_{HL_2} matrix to get a resulting HL'_2 matrix. The inverse Schur operation is shown in equation (29),

$$U_{HL_2}\ \text{x}\ S_{HL_2}\ \text{x}\ U^T_{HL_2}\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (29)$$

10) the inverse DWT is applied on the modified matrix HL'_2. This will then give the final watermarked video F.
11) the video frames f are converted from YUV back to RGB matrix formats
12) the frames will then be reconstructed into the final watermarked video scene F
13) the watermarked scenes are reconstructed to get the final watermarked VXM

the watermark embedding procedure stated from steps 1 to 13 is depicted in Figure 28.

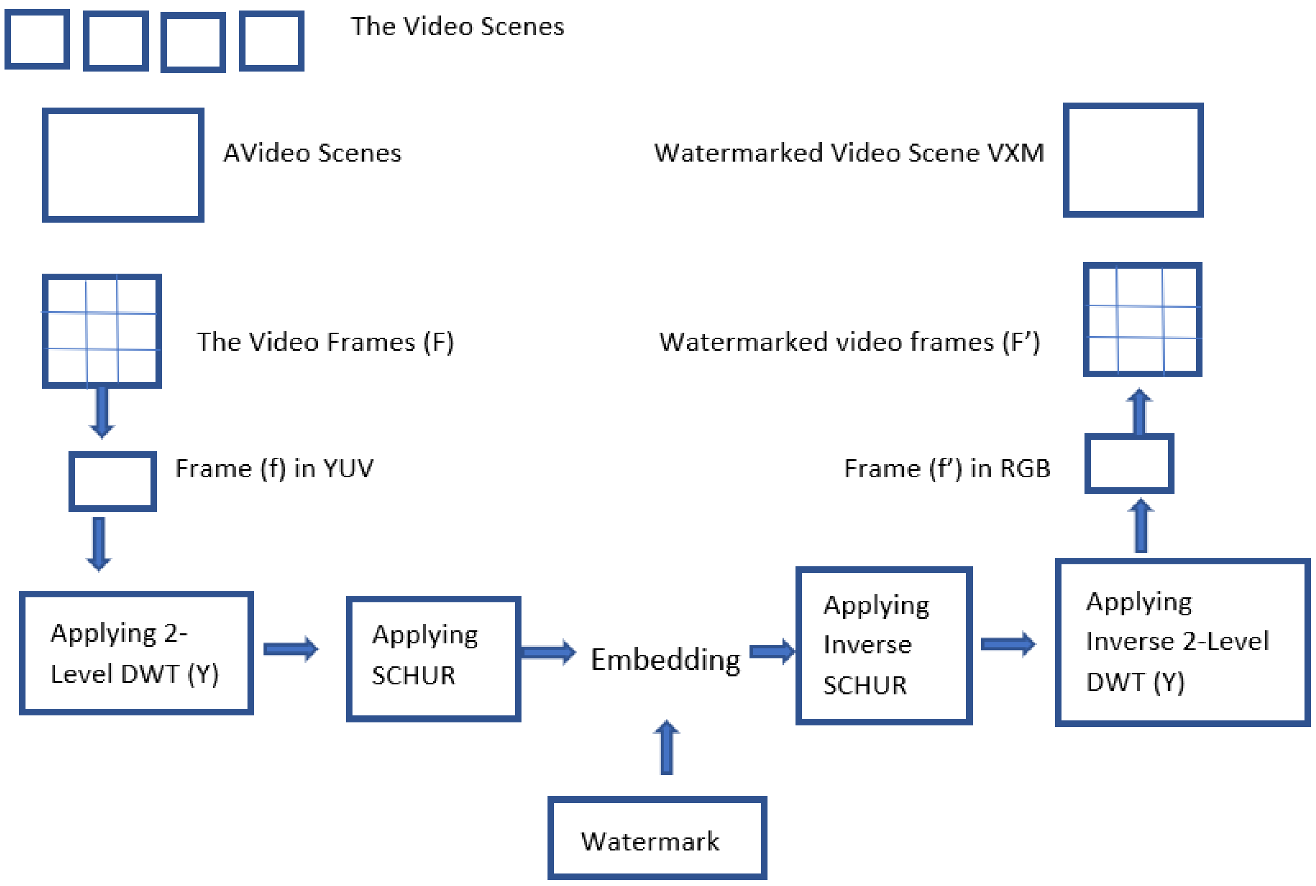


Figure 28 The Watermark Embedding Procedure

II. Watermark Extracting Procedure

Extracting the watermark doesn't need the original video because the proposed DWT-SCGUR algorithm is a blind algorithm. This is why the watermarked image will be extracted straight from the watermarked video frames from the Least Significant Bits directly. A detailed watermark extraction methodology is explained from steps 1 to 9. A Block Diagram showing the basic procedure is shown in Figure 29.

(The Input)

1) The watermarked Video Clip V' is used as the input
2) The watermarked video clip V' is divided into watermarked scenes V'_{si}
3) The watermarked frames of each watermarked video scene are processed using DWT and SCHUR which are described from steps 4 to 7
4) The video Frame F' is converted from RGB matrix to YUV

(Application of 2-Level DWT)

5) The 2-Level DWT for frame F' is calculated. The seven sub-bands produced after this will be $[\mathbf{wLL_1}, \mathbf{wLL_2}, \mathbf{wHL_2}, \mathbf{wLH_2}, \mathbf{wHH_2}, \mathbf{wLH_1}, \mathbf{wHH_1}]$

(Application of SCHUR)

6) The SCHUR operator is applied to the $\mathbf{wHL_2}$ sub-band. The SCHUR operator then decomposes the sub-bands coefficient into two independent matrices, as shown in equation (30),

$$\mathbf{W_{HL_2}} \text{ x } \mathbf{U_{WHL_2}} \text{ x } \mathbf{S_{WHL_2}} \text{ ………………………… } \mathbf{(30)}$$

(The final Extraction)

7) The embedded watermark is extracted from the diagonal matrix $\mathbf{S_{WHL_2}}$ [shown in equation (31)]:

$$\mathbf{W_{V_{si}}(i) = LSB\ (\boldsymbol{S}_{HL_2}\ (\boldsymbol{i}, \boldsymbol{i}))} \text{ ………………………… } \mathbf{(31)}$$

(Reconstructing the Video)

8) The Image Watermark $\mathbf{W_{V_{si}}}$ is constructed by cascading all watermark bits extracted from every frame
9) The same procedure is repeated for every video scene

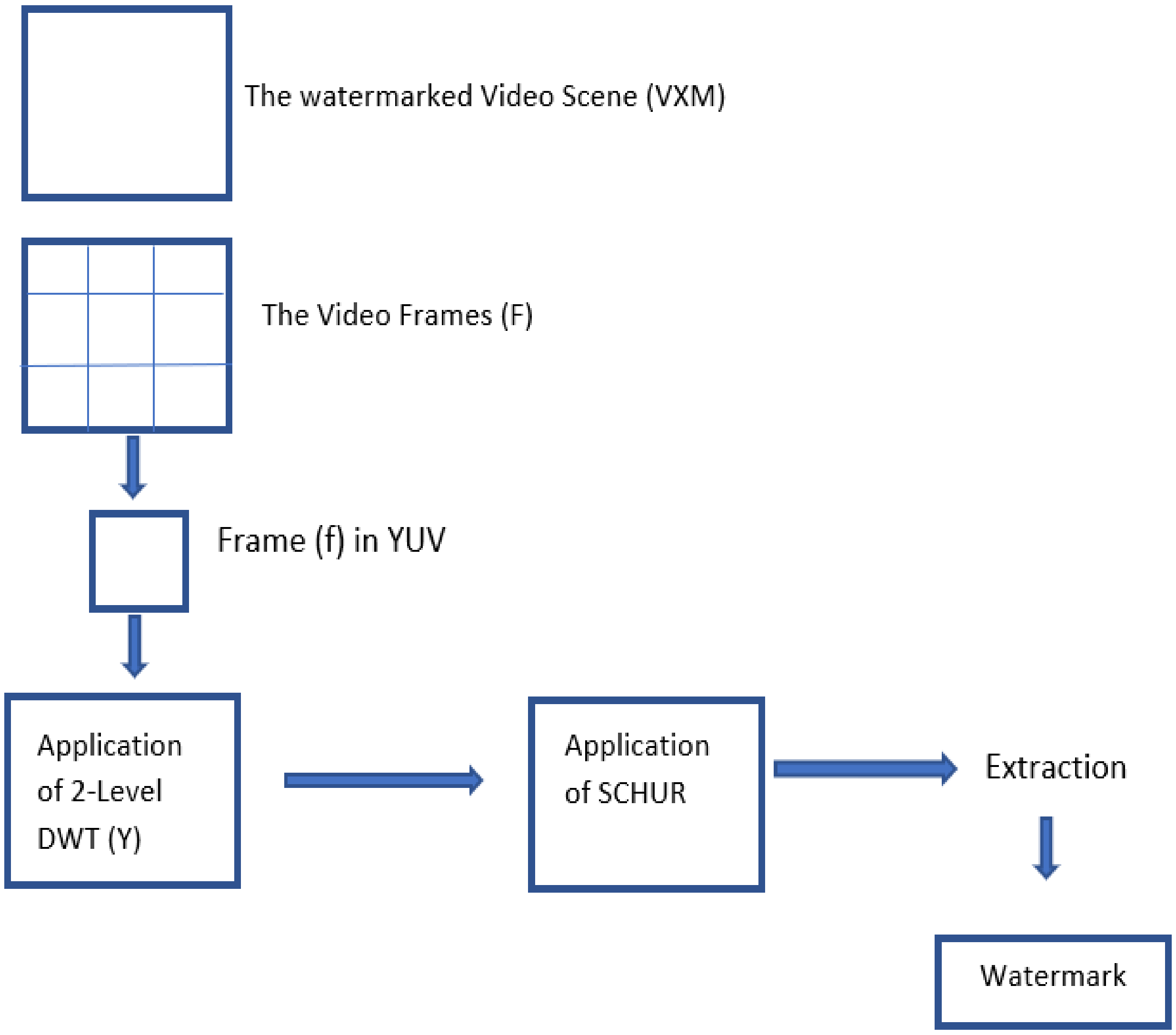


Figure 29 The Watermark Extraction Procedure

3.3) Discrete Cosine Transform (DCT) Digital Video Watermarking (not selected)

The Discrete Cosine Transform Digital Watermarking method is somewhat similar to the spatial domain watermarking with the difference being in the place of changing the image bit plane pixel LSB, the frequency coefficients will be changed instead. The Discrete Cosine Transform is also robust against attacks like compression, filtering, noise, and sharpening. It is based on the general compression of JPEG and is also suitable for watermarking.

The Discrete Cosine Transform allows an image to be decomposed into various frequency bands, helping a watermarking algorithm using Discrete Cosine Transform to easily embed watermark information into the frequency bands of the image(s). for the embedding procedure, the middle-frequency bands are generally preferred, as stated it will increase the robustness against attacks that might distort higher frequencies. In addition to this by choosing the middle frequencies, the important parts of the image that are generally concentrated in the lower frequencies are ignored.

To prevent any extraction of the “hidden” information from the transformed domain directly, the watermarks will be embedded by modifying the neighboring blocks of the middle-frequency coefficients of the original image instead of using an additive operation.

The Discrete Cosine Transform is based on dividing the original image into an 8 x 8 image pixel block(s). the middle-frequency range from the DCT coefficients is extracted with the help of a 2D middle frequency mask.

If at all the assumption is made that the digital watermark is a binary image, it will be permutated by a fast 2D pseudo-random traversing method. Resulting in a spatial relationship that is dispersed to resemble a noise. The block-based permutations and the pixel-based permutations are applied here.

The watermark extraction procedure will use the original cover image as well as the watermark. Initially, the Discrete Cosine Transform must be performed on the original and the watermarked image(s). then the original cover image will be subtracted from the watermarked image(s). the pixel-based permutations and the block-based permutations need to be reversed to get the extracted watermark.

I. ***A DCT/DWT Video Watermarking Technique (comparison)***

In this method, the watermarking concept is used to embed the data (image) using both the DCT (explained in Chapter 6, 3.3) and DWT (explained chapter 5, 3.1). the steps which have been used to embed the data (image) into the video frames are:

(At the Input Video Block)

1) The video that needs to be watermarked is selected from the multimedia file.
2) The selected media will then be converted into respective frames
3) The respective frames will then be converted into an unsigned integer format
4) Using the selector block the video will be selected, using this block the block size can be selected where the chosen data needs to be embedded
5) The video selected will then be pre-processed using the 2D-DCT/DWT block processing unit
6) The resulting output will be given to a matrix block, and the video can be observed using the video viewer.

(At the Image Block)

7) Using the file block the image to be embedded can be selected
8) The image will then be converted into an unsigned integer data format
9) The data output or the message will be selected using the selector block
10) The selected message will be processed using 2D-DCT/DWT using the block processing block
11) The output is given as the input to the matrix sum block

(The Embedding Process)

12) The image will be embedded in the video (in the matrix sum block) which needs to be watermarked
13) The resulting image embedded video frame is given to the block processing unit where it will undergo 2D-DCT/DWT processing
14) The processed result is converted to a single data type which again will be connected to the input terminal of the video viewer to observe the result.

A block diagram depicting the above steps is shown in Figure 30. The equation for the 2D- DCT (Rahman, June 2013) is shown in equation (32):

$$\mathbf{F(m,n)} = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} C(m)C(n)f(i,j)\cos\left[\frac{\pi(2i+1)m}{2N}\right] * \cos\left[\frac{\pi(2j+1)m}{2N}\right] \ldots\ldots\ldots\ldots\ldots\ldots \textbf{(32)}$$

Where n is the level number, (i,j) is the pixel number, N is the number of frames, and f(i,j) is the inverse DCT

C(m),C(n) is $\sqrt{1/N}$ if m,n is 0 and $\sqrt{2/N}$ if m,n is 1 up to N-1.

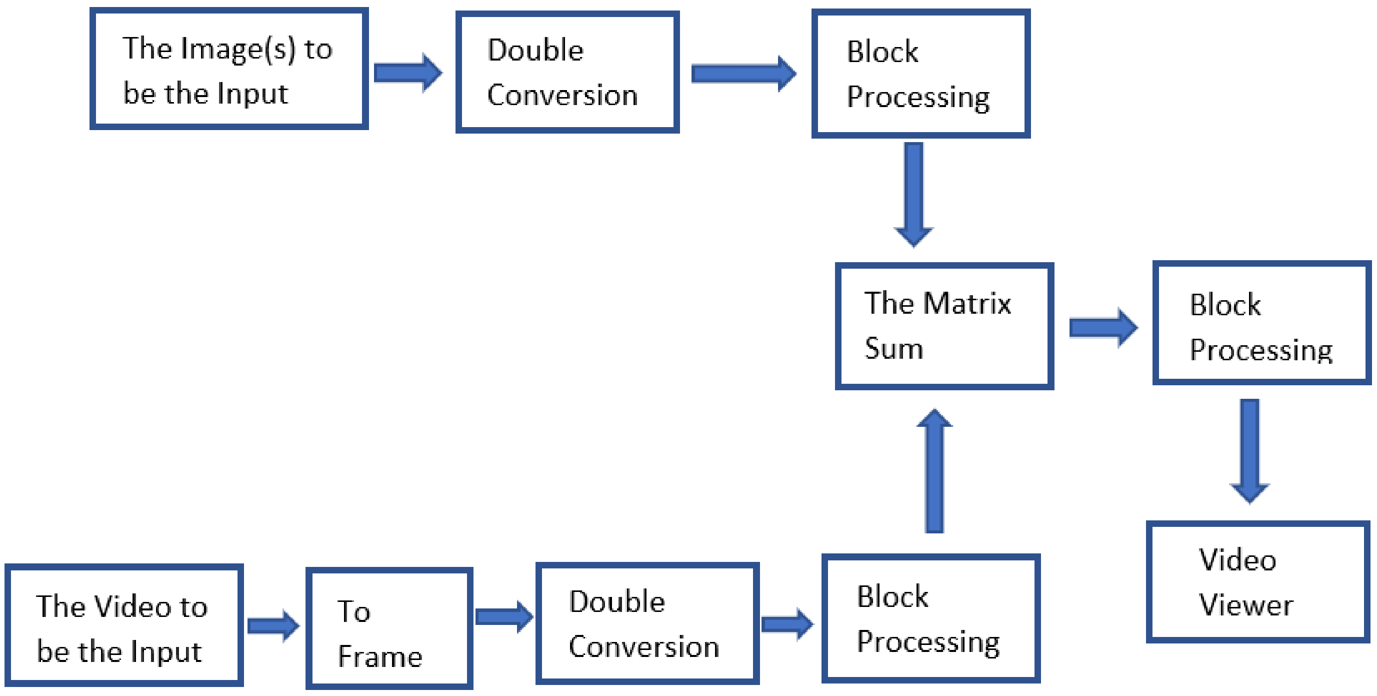


Figure 30 The Block Diagram for the Video Watermarking using DCT/DWT Algorithm

In the report published in (Shan, January 2015), the simulation results (using SSIM and PSNR) were observed wherein it was observed that the results from the DWT algorithm were giving better results when compared to the DCT for the digital watermarking on a grayscale sequence. And, against various attacks, the DWT method was providing better protection and results again when compared to that of the DCT method. Meaning the DCT technique did not show as promising results as the DWT did and so the DCT algorithm is not as good as DWT based on the results (Shan, January 2015) which is why the DCT-SCHUR method was not chosen for digital video watermarking.

3.4) <u>DWT-DCT-SVD based Digital Video Watermarking (not selected)</u>

this algorithm uses a combined method of the Discrete Wavelet Transform (explained chapter 5, 3.1), the Discrete Cosine Transform (explained in Chapter 6, 3.3), and Singular Value Decomposition (SVD). the SVD matrix is quite helpful and a very important tool in the image(s) transformation. The SVD of a given image "F" is defined as equation (33),

$$\mathbf{F} = \mathbf{USV}^T \text{ .... } \mathbf{(33)}$$

Where S is the diagonal matrix (Rahman, June 2013), shown in equation (34),

$$\mathbf{S} = \begin{matrix} s_1 & \cdots & \mathbf{0} \\ \vdots & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & s_n \end{matrix} \text{ .... } \mathbf{(34)}$$

The terms V and U are the orthogonal matrices shown in equation (35),

$$\mathbf{V}^T * \mathbf{V}^T = \mathbf{U}^T * U = \mathbf{1} \text{ .... } \mathbf{(35)}$$

The block diagram(s) referring to the DWT-DCT-SVD algorithm in (Rahman, June 2013), is:

I. ***Embedding the Watermark***

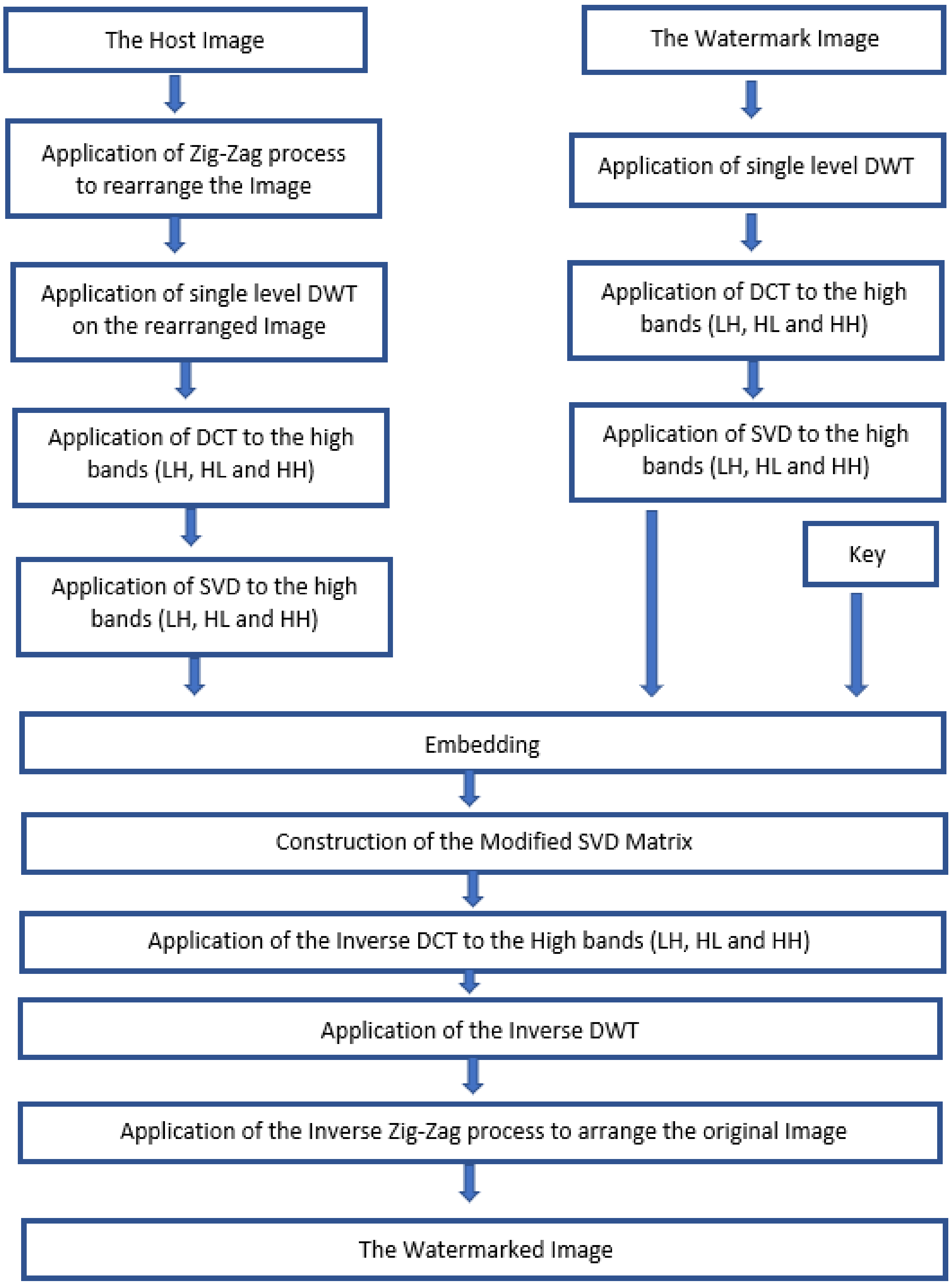
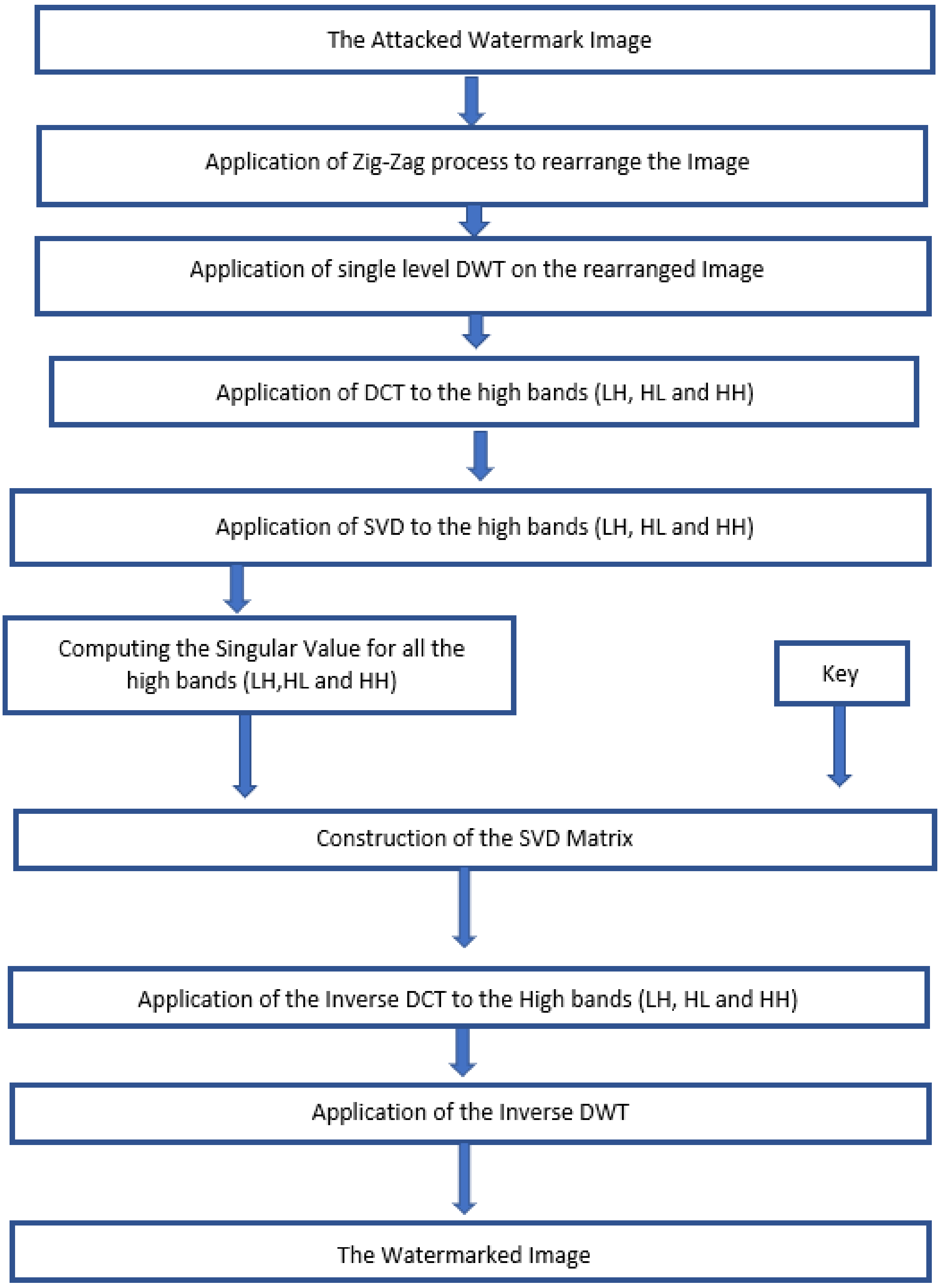

The Host Image
Application of Zig-Zag process to rearrange the Image
Application of single level DWT on the rearranged Image
Application of DCT to the high bands (LH, HL and HH)
Application of SVD to the high bands (LH, HL and HH)
The Watermark Image
Application of single level DWT
Application of DCT to the high bands (LH, HL and HH)
Application of SVD to the high bands (LH, HL and HH)
Key
Embedding
Construction of the Modified SVD Matrix
Application of the Inverse DCT to the High bands (LH, HL and HH)
Application of the Inverse DWT
Application of the Inverse Zig-Zag process to arrange the original Image
The Watermarked Image

Figure 31 The Watermark Embedding Algorithm used in (Rahman, June 2013)

II. ***Extracting the Watermark***



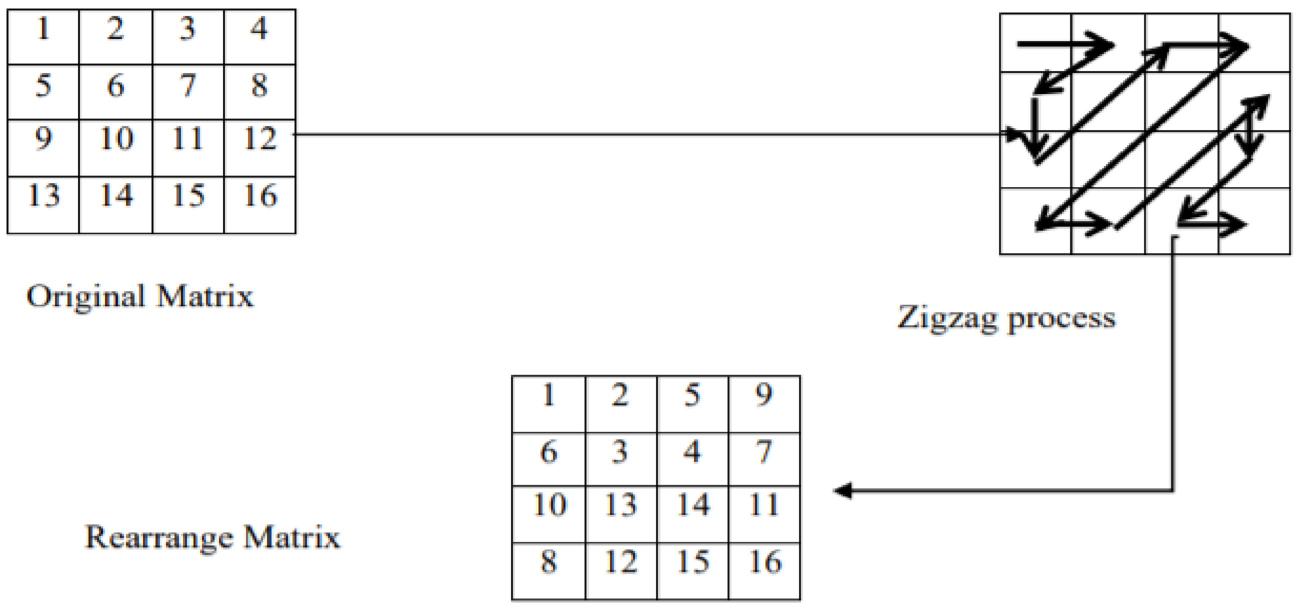
The Attacked Watermark Image
Application of Zig-Zag process to rearrange the Image
Application of single level DWT on the rearranged Image
Application of DCT to the high bands (LH, HL and HH)
Application of SVD to the high bands (LH, HL and HH)
Computing the Singular Value for all the high bands (LH,HL and HH)
Key
Construction of the SVD Matrix
Application of the Inverse DCT to the High bands (LH, HL and HH)
Application of the Inverse DWT
The Watermarked Image

Figure 32 The Watermark Extraction Algorithm used in (Rahman, June 2013)

III. <u>Block Diagram Explanation</u>

(In the Watermark Embedding Algorithm)

1) The Host Image (HI) is used as the input
2) The Host Image (HI) is rearranged by the application of a zig-zag scanning process to obtain a RI (the Rearranged Image)



1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

1	2	5	9
6	3	4	7
10	13	14	11
8	12	15	16

Figure 33 Rearranging the Matrix by the said Zig-Zag Process (Rahman, June 2013)

3) The Single Level DWT is applied on the Rearranged Image (RI) to help decompose it to four sub-bands **LL, HL, LH, HH.**
4) All the high-bands are selected (the LL, HL, HH) of the Rearranged Image (RI). The DCT will then be applied to these high bands as well.

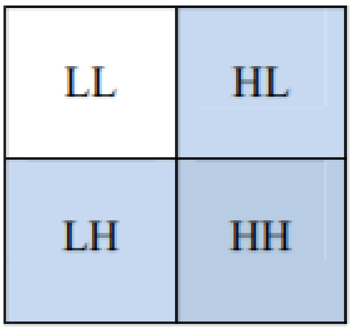


Figure 34 The selected high-bands (Rahman, June 2013)

5) The Singular Value Decomposition (SVD) is then applied to these high bands to obtain the bands SH1, Sh2, and SH3.
6) The watermark image (wi) is used as input. The single-level DWT is then applied to help decompose it in 4 sub-bands namely the LL1, HL1, LH1, HH1.
7) The high bands (LH1, HL1, HH1) are selected from the input watermark image (wi) and DCT is applied to all these high bands.
8) From there SVD is applied to the high bands LH, HL, HH to obtain SW1, SW2, SW3.
9) The SH1, SH", SH3 are modified with the help of the equation $S_{wi} = S_i + \alpha + S_w$, where the value of i is from 1 to 3.

10) The modified SVD matrix (LH11, HL11, HH11) is constructed.
11) The inverse DCT is applied to all high bands LH11, HL11, HH11, and the inverse DWT is applied with LL.
12) The inverse Zig-Zag procedure is applied to arrange the original position of the image and the final watermarked image (WI) is obtained.

(In the Watermark Extraction Algorithm)

13) Th watermarked image (WI) is the input
14) The watermarked image(Wi) is re-arranged with the help of the Zig-Zag process to obtain a rearranged image RI*.
15) The single-level DWT is applied to the rearranged image RI* which will decompose it into 4 sub-bands: LL*, HL*, LH*, HH*.
16) The high bands of RI* are selected (LH*, HL* and HH*), the DCT will then be applied to these bands.
17) Then the SVD will also be applied to these bands to get SH1*, SH2*, SH3*
18) The SH1*, SH2*, SH3* are modified using the equation $S_w = \ (S_{wi} - S_i)/\ \alpha$, where the value of i is from 1 to 3.
19) The modified SVD matrix (LH1*, HL1*, HH1*) is constructed
20) The inverse DCT is applied to the high bands LH1*, HL1*, HH1*.
21) The inverse DWT is applied to all the bands to get them out as the watermark image.

From the journal published in (Rahman, June 2013), it can be observed that this algorithm may be one of the best watermark embedding and extraction algorithms out there, as it combines the secure nature and benefits of the Discrete Wavelet Transform (DWT0, the Discrete Cosine Transform (DCT) and the Singular Value Decomposition (SVD) into one bulk algorithm, this algorithm is believed to provide better protection than the proposed DWT-SCHUR algorithm (Chapter 6, 3.2), but unfortunately this algorithm is quite bulky and it is not recommended to run alongside any other program, algorithm and/or software programming code.

The three programs (proposed) when run together do tend to heat up and stress the hardware, due to the hardware limitations running the proposed DWT-DCT-SVD algorithm may not give the proper results or may damage the hardware, which is why the DWT-SCHUR algorithm was chosen, it put less stress on the hardware and also gave amazing results which when compared to the DWT-DCT-SVD results do not have any difference when seen by the naked eye.

Chapter 7: Results & Discussions

The entire codes for all three programs (the Detection and Tracking of a moving object, the Video quality enhancement, and the video watermarking using Discrete Wavelet Transform) has been implemented in the MATLAB software. The hardware consists of a dual-core Intel processor at 2.4GHz which can process around 50 to 60 frames each second. The Detection of the moving object(s) that has been performed on the system has been on a video of moving vehicles under a few different kinds of evaluation (the morning, the afternoon, and the evening). For evaluation purposes the three videos will be run by the program code, the results saved in the hardware, a remote alert send to the mail and the results will be shown in Figure 35, Figure 36, Figure 37, and Figure 38.

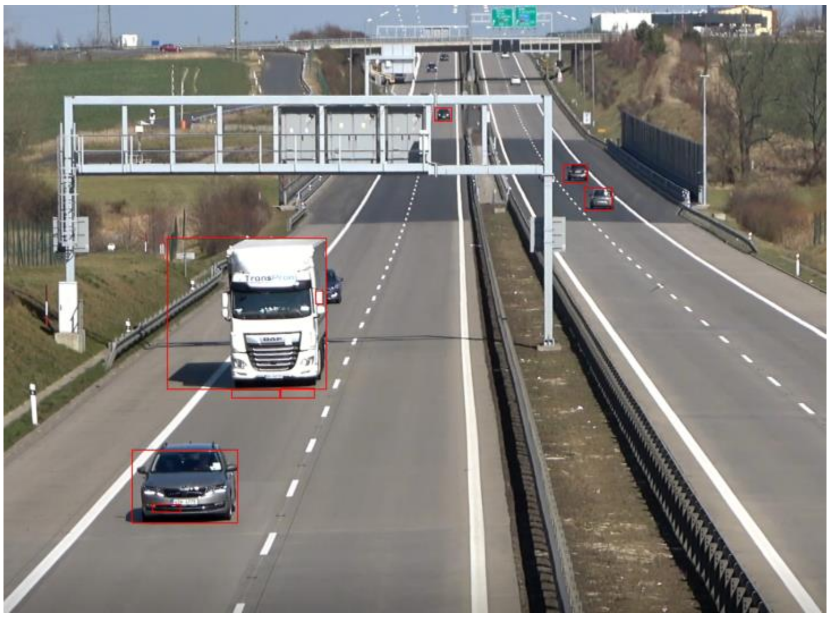


Figure 35 Result of the Object Detection and Tracking for Vehicles in the Afternoon with the Objects Moving in two directions (Bidirectional) [Sample Video 1]

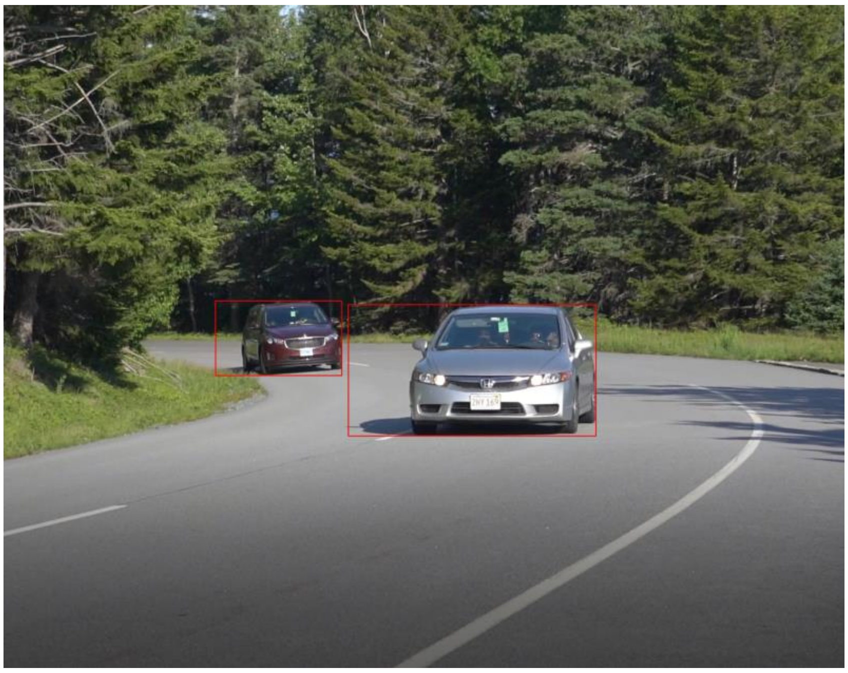


Figure 36 The Result of Object Detection and Tracking in the Morning with objects moving in one direction (Unidirectional) [Sample Video 2]

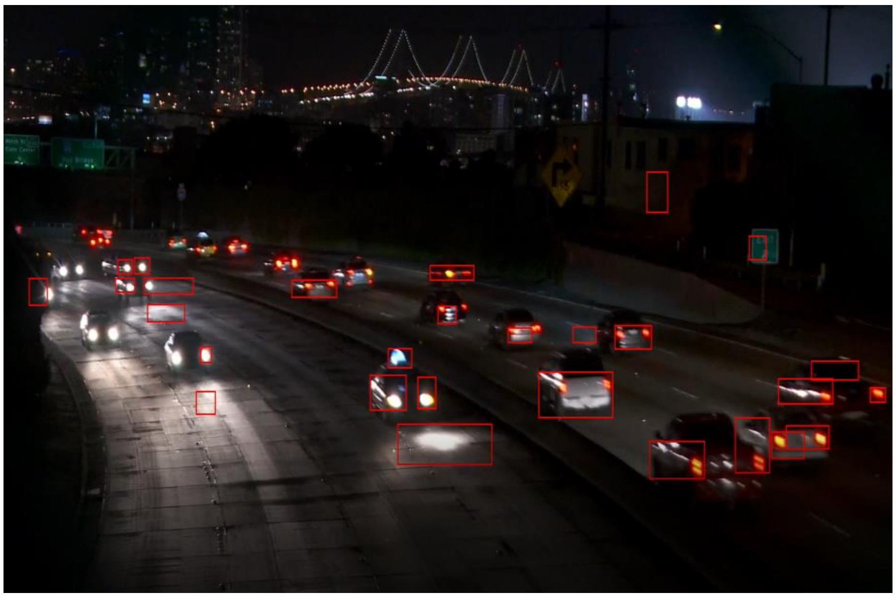
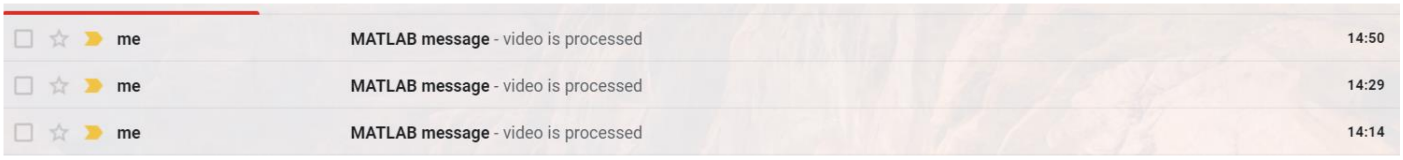


Figure 37 The Result of Object Detection and Tracking in the Night with Objects moving in two directions (Bidirectional) [Sample Video 3]



me MATLAB message - video is processed 14:50

me MATLAB message - video is processed 14:29

me MATLAB message - video is processed 14:14

Figure 38 The remote alerting send by MATLAB to the email domain of the selected user for the three videos processed

In Figure 35 and Figure 36, it can be seen that the objects are detected properly with minimum noise in both unidirectional and bidirectional cases, however, in Figure 37 it can be seen that the MATLAB software seems to be confused when comparing the moving object(s) (the moving cars) from the moving light reflections on the road, and also due to the cluttered objects in each frame with the current processing hardware it seems to be unable to capture every moving object in every frame. From plain sight, it can be said that the ideal results would be obtained from a scene which contains high illumination (such as during day time and afternoon time) and it is not recommended to expect good results during the night time, and also if the scene has less number of objects. For the noise part, it is recommended to try different morphological operations for the first frame to observe which has the least amount of noise. For the inability to detect objects present in a cluttered scene one option is to increase the MinimumBlobArea from 200 to about 1000 or 2000 depending on the hardware. The change in the MinimumBlobArea will only help to track objects that are far away and as such will not play a big role improving the object detection in a cluttered environment, it will however detect the far away objects that previously were too small to be detected as objects making sure more number of objects are detected. To detect the objects clearly in a cluttered scene in addition to choosing the correct morphological operation it is highly recommended to use a Convolution Neural network, unfortunately, the hardware used in this report does not meet the specifications to either create or read the data from and to a Convolution Neural Network (CNN) (Shuiwang Ji, 2013) and MATLAB has also discontinued the use of many functions and commands to operate the neural networks, the only way to use a CNN (Convolution Neural Network) is to export the code from MATLAB to JAVA or OpenCV.

For the video(series of Images) quality enhancement, this report will currently use the output whose screenshot is shown in Figure 36. The frame of interest is shown in Figure 39, and the corresponding enhanced frame is shown in Figure 40.

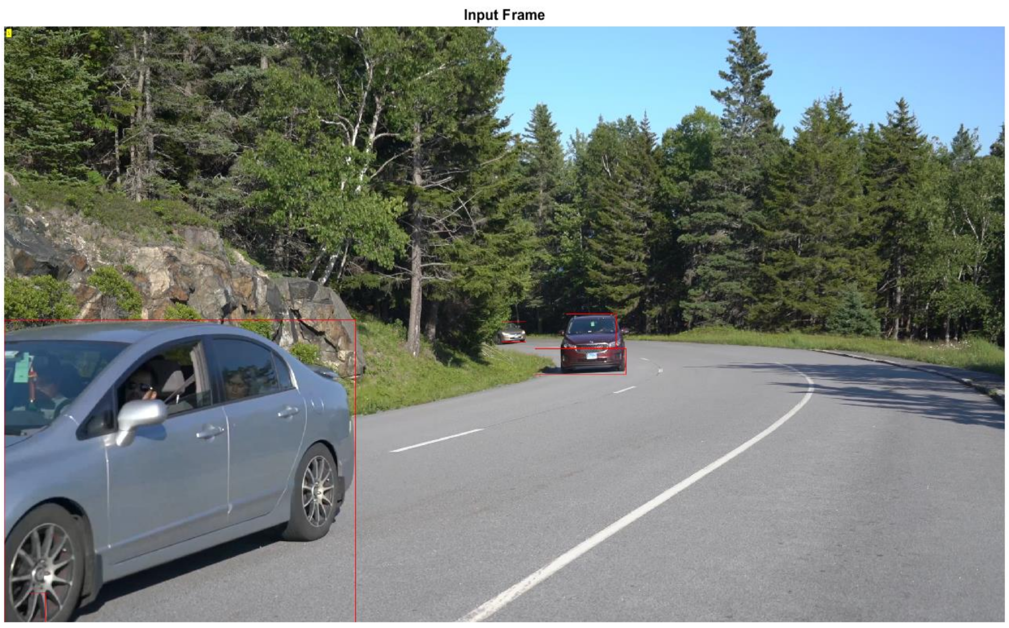


Figure 39 The Frame of the Output Video before the Image(s) Enhancement Algorithm is applied

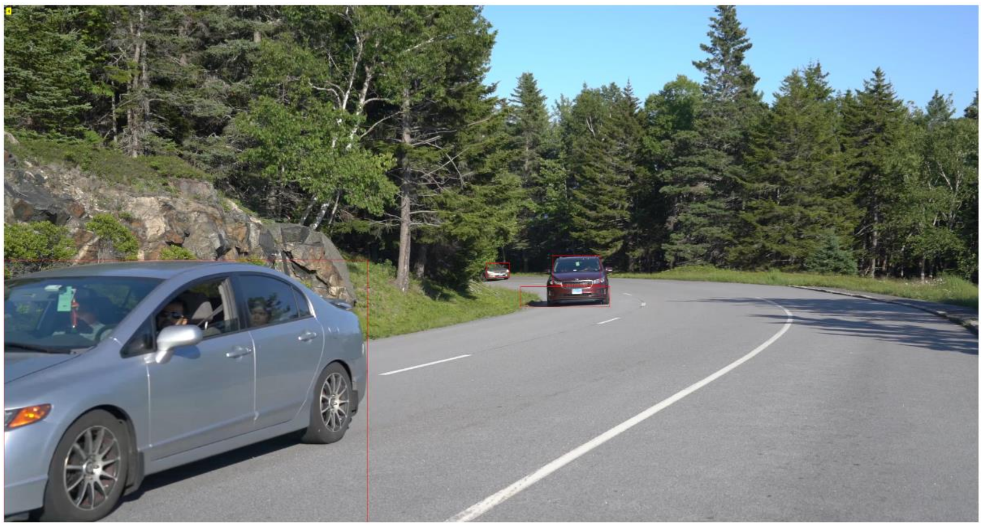


Figure 40 The same Frame captured in Figure 39 after the application of the Image(s) Enhanced Algorithm

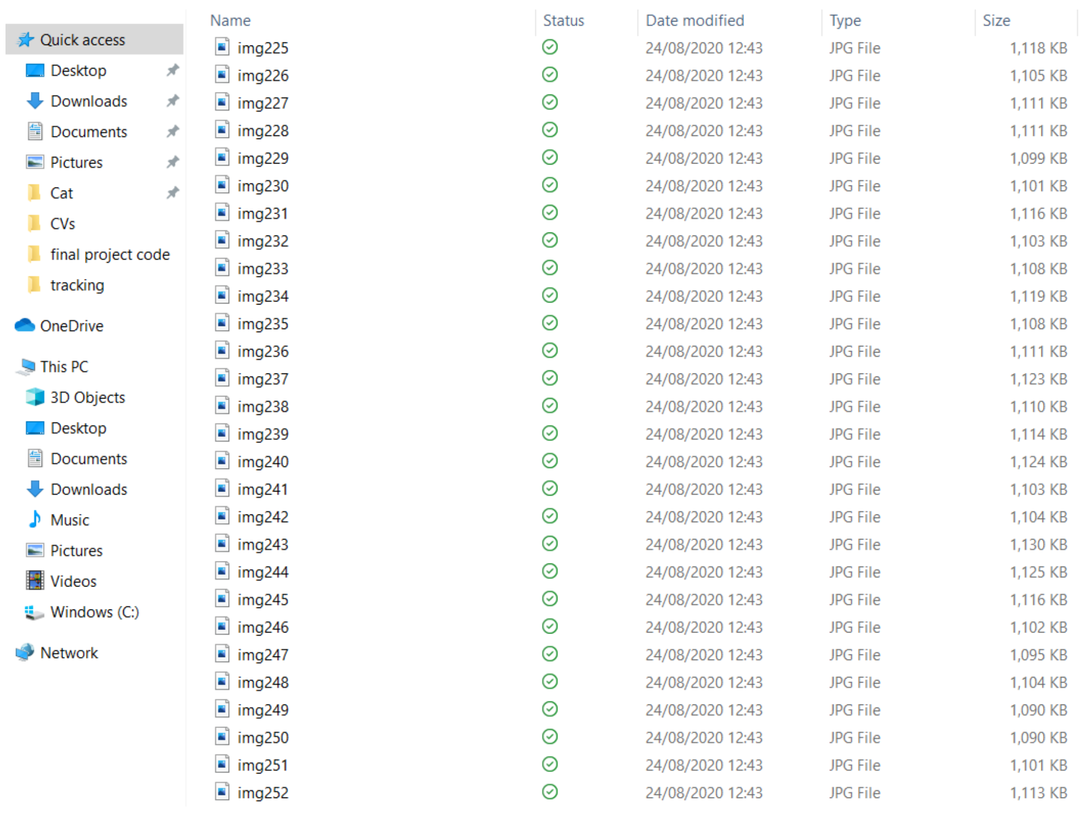
The entire output of the said video which was the output of the video used in Figure 36 will be broken down to frames and each frame will be enhanced for its contrast and its deblurring.

The video used as the input was over 6 seconds long and it contained approximately 252 frames which have each been stored in the file in the hardware as seen in Figure 41. Due to the hardware being a commercial one it is advisable to use video inputs that are around 6 to 8 seconds long to prevent the MATLAB software to run out of internal memory to process the frames and to prevent pressurizing the hardware. If the user wishes to use a video that is very long there are two methods:

1) To use a high-end server desktop PC with high RAM to help the MATLAB software process more frames and also process them faster.
2) Since this report was kept keeping the casual store owners, jewelry shops small-time business, and large time businesses to save costs this step can be completely ignored and the user may directly go the next stage (Video Watermarking using Discrete Wavelet Transform) without even considering the enhancement of the image frames (a cheaper alternative but not recommended as the camera that

captures these footages may be of low pixel quality, If it is a good quality camera this step is completely optional).

In Figure 40 the output frame after image enhancement is not that different when compared to the frame used before the image enhancement algorithm in Figure 39, as the video obtained from the source (stationary camera) was already of high quality. The saved frames shown in Figure 41 will be played in one motion after the video has been processed.



Name	Status	Date modified	Type	Size
img225	✓	24/08/2020 12:43	JPG File	1,118 KB
img226	✓	24/08/2020 12:43	JPG File	1,105 KB
img227	✓	24/08/2020 12:43	JPG File	1,111 KB
img228	✓	24/08/2020 12:43	JPG File	1,111 KB
img229	✓	24/08/2020 12:43	JPG File	1,099 KB
img230	✓	24/08/2020 12:43	JPG File	1,101 KB
img231	✓	24/08/2020 12:43	JPG File	1,116 KB
img232	✓	24/08/2020 12:43	JPG File	1,103 KB
img233	✓	24/08/2020 12:43	JPG File	1,108 KB
img234	✓	24/08/2020 12:43	JPG File	1,119 KB
img235	✓	24/08/2020 12:43	JPG File	1,108 KB
img236	✓	24/08/2020 12:43	JPG File	1,111 KB
img237	✓	24/08/2020 12:43	JPG File	1,123 KB
img238	✓	24/08/2020 12:43	JPG File	1,110 KB
img239	✓	24/08/2020 12:43	JPG File	1,114 KB
img240	✓	24/08/2020 12:43	JPG File	1,124 KB
img241	✓	24/08/2020 12:43	JPG File	1,103 KB
img242	✓	24/08/2020 12:43	JPG File	1,104 KB
img243	✓	24/08/2020 12:43	JPG File	1,130 KB
img244	✓	24/08/2020 12:43	JPG File	1,125 KB
img245	✓	24/08/2020 12:43	JPG File	1,116 KB
img246	✓	24/08/2020 12:43	JPG File	1,102 KB
img247	✓	24/08/2020 12:43	JPG File	1,095 KB
img248	✓	24/08/2020 12:43	JPG File	1,104 KB
img249	✓	24/08/2020 12:43	JPG File	1,090 KB
img250	✓	24/08/2020 12:43	JPG File	1,090 KB
img251	✓	24/08/2020 12:43	JPG File	1,101 KB
img252	✓	24/08/2020 12:43	JPG File	1,113 KB

Figure 41 The Frames obtained from the Video obtained after the Object Detection and Tracking, stored within the hardware Frame wise

Once the image(s) enhancement processing has been taken care of, the next and the final step in this report will be the embedding and the extraction of a watermark using DWT-SCHUR and discussing the robustness against different types of attacks. For the video used in this report for this case, there will be four scenes chosen, and a shot of these scenes are shown in Figure 42. The watermark is a binary format logo shown in Figure 43. The use of the watermarks makes it easy to test the robustness of the watermarking technique. Here put simply the extracted logo will be compared to the embedded logo.

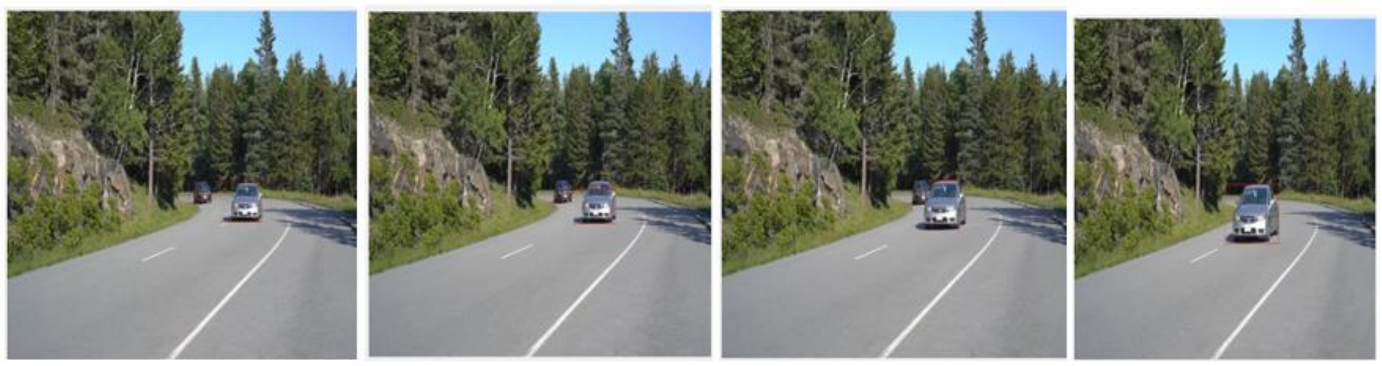


Figure 42 Some of the Frames captured from the processed Video before Embedding the Watermark

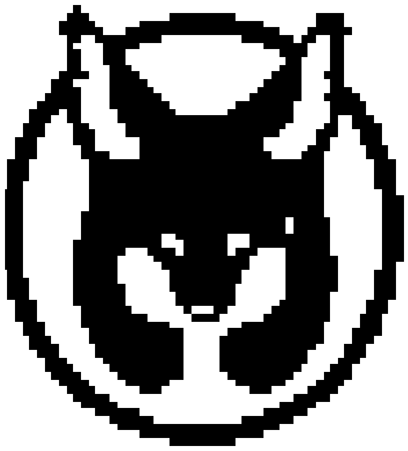


Figure 43 The Watermarked Binary Logo

After the application of the proposed technique, it was observed that there was no visual degradation in the video formats after embedding the watermarked logo and the similarity present in between the extracted and the embedded images are completely similar as seen in Figure 44 (for the eye comparison of the extracted and the embedded binary image) and Figure 45 (for the output frames of the video after being embedded with the watermarked image). The average value for the PSNR for the first 64 watermarked frame(s) is shown in Figure 46 and their respective values are shown in Figure 48. At this value of the PSNR, there was no degradation in the quality of the watermarked video. The equation used for the PSNR (peak signal to noise ratio) is shown in equation (36), while the correlation between the embedded and the extracted watermark is 1 as seen in Figure 47.

$$\mathbf{PSNR = 10 * Log_{10}(}^{\mathbf{255}}/_{\sqrt{\mathbf{MSE}}}) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \textbf{(36)}$$

Where MSE (the mean square error) is shown in equation (37),

$$\mathbf{MSE} = \frac{\sum_{M,N}[\,\mathbf{I_1}(m,n)-\mathbf{I_2}(m,n)]^{\,2}}{\mathbf{M*N}} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \textbf{(37)}$$

Where M and N are the numbers of rows and columns in the input images.

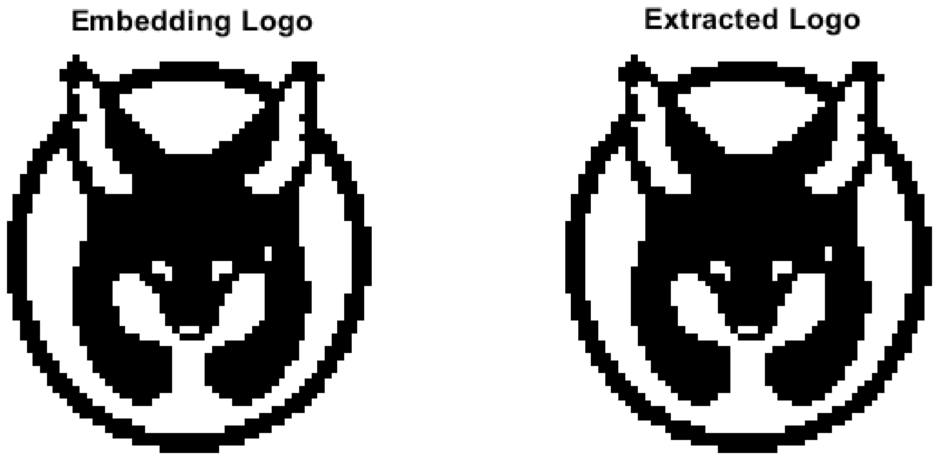


Figure 44 The Original (embedded) Watermark Logo [during Embedding] and the Extracted Watermark Logo [after Extracting]

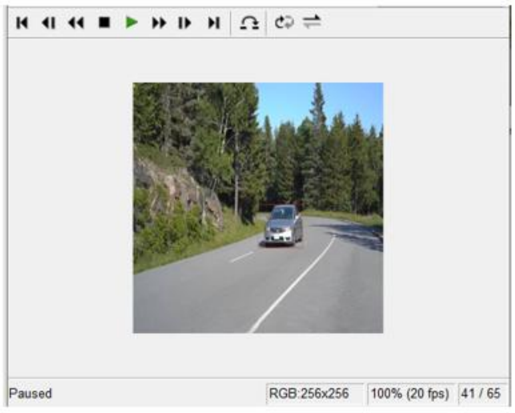


Figure 45 The Watermark Embedded Video clip (top) the Video Frames of the Embedded Video (bottom)

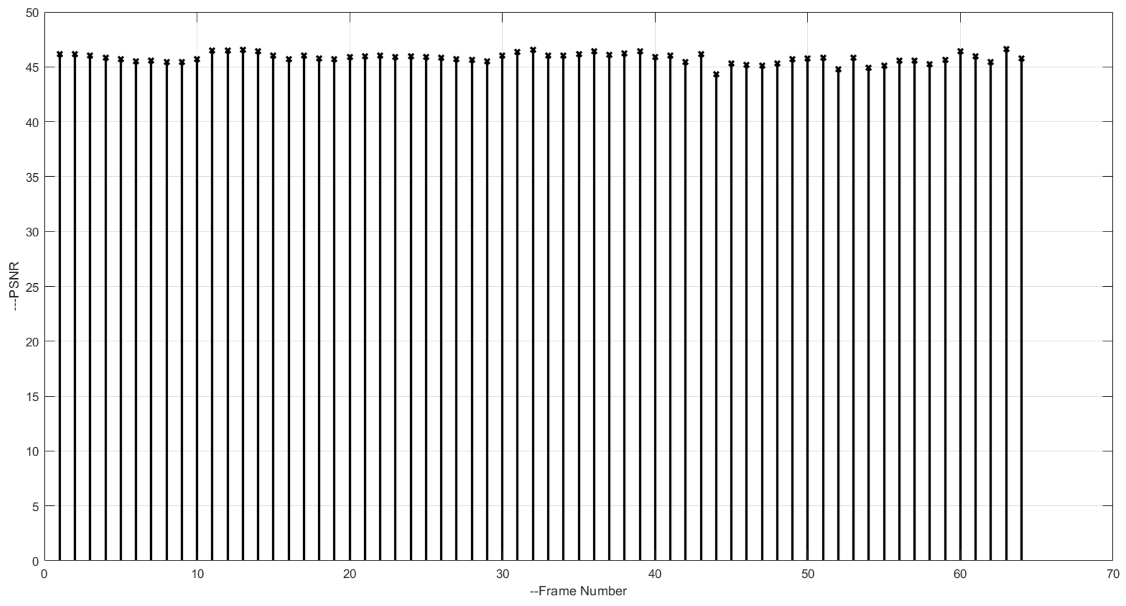
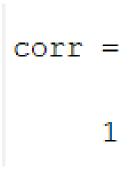


Figure 46 The PSNR values for the first 64 Watermarked Frames



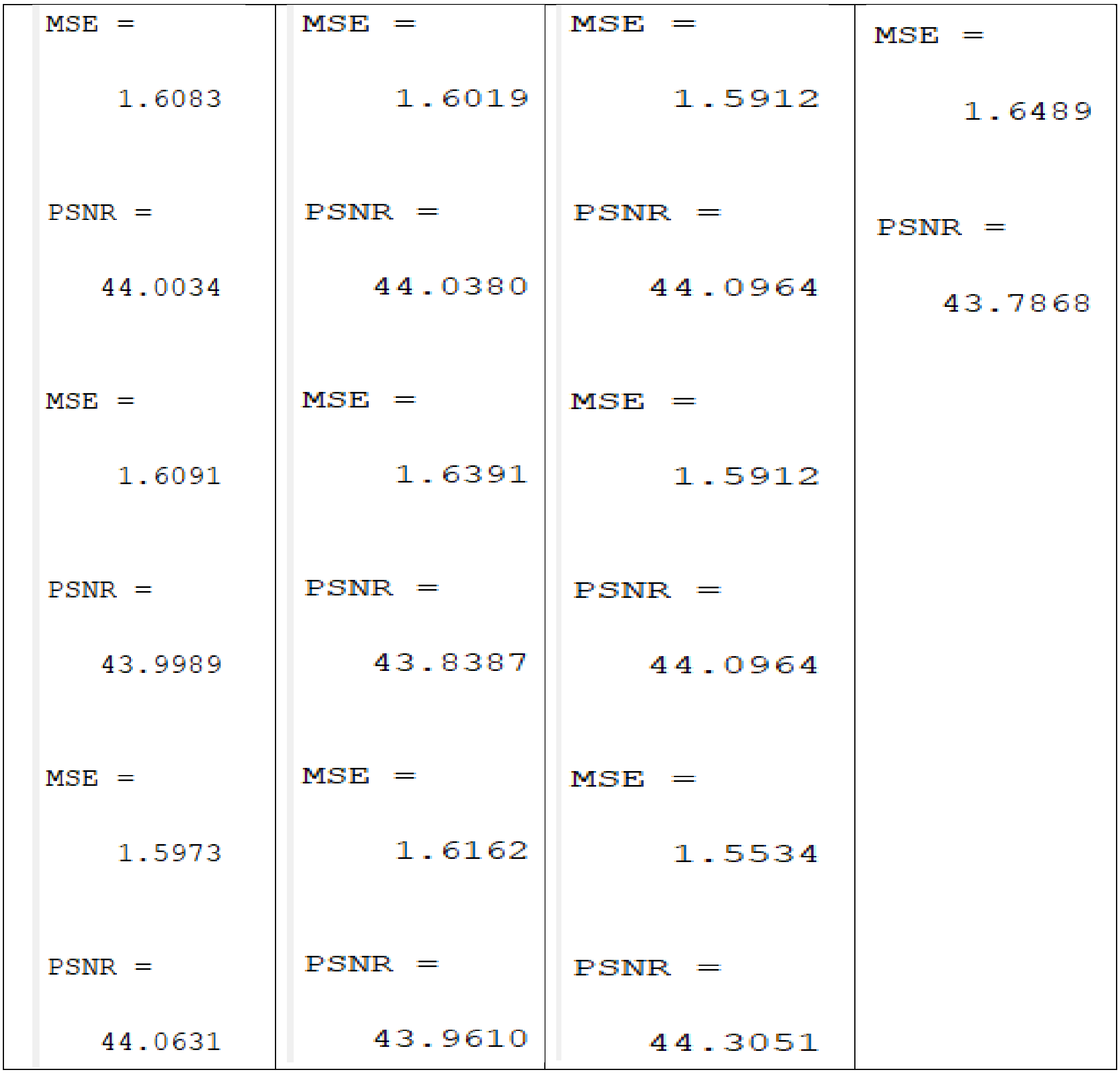
```
corr =

      1
```

Figure 47 The Correlation between the Extracted and the Embedded Watermark without any noise (using equation 38)

MSE =	MSE =	MSE =	MSE =	MSE =	MSE =
1.5370	1.5499	1.5034	1.5137	1.4342	1.5105
PSNR =	PSNR =	PSNR =	PSNR =	PSNR =	PSNR =
44.3971	44.3249	44.5893	44.5301	44.9988	44.5483
MSE =	MSE =	MSE =	MSE =	MSE =	MSE =
1.5370	1.5253	1.5001	1.5568	1.4349	1.5032
PSNR =	PSNR =	PSNR =	PSNR =	PSNR =	PSNR =
44.3971	44.4636	44.6087	44.2861	44.9942	44.5906
MSE =	MSE =	MSE =	MSE =	MSE =	MSE =
1.6136	1.5140	1.4688	1.5790	1.4284	1.4873
PSNR =	PSNR =	PSNR =	PSNR =	PSNR =	PSNR =
43.9750	44.5280	44.7914	44.1629	45.0340	44.6830

MSE =	MSE =	MSE =	MSE =	MSE =	MSE =
1.5869	1.6107	1.5576	1.4928	1.6047	1.6189
PSNR =	PSNR =	PSNR =	PSNR =	PSNR =	PSNR =
44.1196	43.9908	44.2818	44.6505	44.0232	43.9467
MSE =	MSE =	MSE =	MSE =	MSE =	MSE =
1.5944	1.5346	1.5586	1.5311	1.6047	1.5727
PSNR =	PSNR =	PSNR =	PSNR =	PSNR =	PSNR =
44.0788	44.4109	44.2763	44.4309	44.0232	44.1978
MSE =	MSE =	MSE =	MSE =	MSE =	MSE =
1.5886	1.5954	1.5038	1.5298	1.5865	1.5132
PSNR =	PSNR =	PSNR =	PSNR =	PSNR =	PSNR =
44.1104	44.0733	44.5870	44.4380	44.1222	44.5330

MSE = 1.5753	MSE = 1.6585	MSE = 1.7307	MSE = 1.5807	MSE = 1.5777	MSE = 1.5712
PSNR = 44.1836	PSNR = 43.7366	PSNR = 43.3662	PSNR = 44.1541	PSNR = 44.1705	PSNR = 44.2062
MSE = 1.5753	MSE = 1.6762	MSE = 1.7300	MSE = 1.5397	MSE = 1.5714	MSE = 1.6054
PSNR = 44.1837	PSNR = 43.6441	PSNR = 43.3700	PSNR = 44.3821	PSNR = 44.2049	PSNR = 44.0190
MSE = 1.6139	MSE = 1.7301	MSE = 1.6357	MSE = 1.5525	MSE = 1.5478	MSE = 1.5885
PSNR = 43.9731	PSNR = 43.3695	PSNR = 43.8568	PSNR = 44.3103	PSNR = 44.3366	PSNR = 44.1112



```
MSE =

    1.6083


PSNR =

   44.0034


MSE =

    1.6091


PSNR =

   43.9989


MSE =

    1.5973


PSNR =

   44.0631
```

```
MSE =

    1.6019


PSNR =

   44.0380


MSE =

    1.6391


PSNR =

   43.8387


MSE =

    1.6162


PSNR =

   43.9610
```

```
MSE =

    1.5912


PSNR =

   44.0964


MSE =

    1.5912


PSNR =

   44.0964


MSE =

    1.5534


PSNR =

   44.3051
```

```
MSE =

    1.6489


PSNR =

   43.7868
```

Figure 48 The PSNR and the MSE values for the first 64 Frames

The results of the other videos for the image(s) or video watermarking using DWT-SCHUR can be found in Appendix A.

The robustness of the proposed method is known to provide security against two types of attacks that being the frame attacks and the standard attacks. The standard attacks can include (but are not limited to) rotation, salt and pepper noise, Gaussian noise, compression amongst various others (Voloshynovskyy, 2001). The video frame attacks can include frame swapping, dropping, and frame averaging. To understand if either of the attacks has happened the similarity between the extracted and the original watermark needs to be measured for similarity using the correlation. The equation for the correlation between the original watermark (A) and the extracted watermark (B) is shown in equation 38, this computes the correlation coefficient.

$$\textbf{Corr2 (A, B)} = \frac{\sum_m \sum_n (\mathbf{A}_{mn} - \overline{\mathbf{A}}) * (\mathbf{B}_{mn} - \overline{\mathbf{B}})}{\sqrt{(\sum_m \sum_n (\mathbf{A}_{mn} - \overline{\mathbf{A}})^2) * (\sum_m \sum_n (\mathbf{B}_{mn} - \overline{\mathbf{B}})^2)}} \quad \textbf{(38)}$$

Where $\overline{\mathbf{A}}$ is the mean2(A) function which computes the mean of all values in array A, and similarly $\overline{\mathbf{B}}$ is the mean2(B) function which computes the mean of all values in array B. The correlation can have the values ranging from 0 (there is absolutely no similarity between the watermark images) to 1 (both the watermark images are the same) as seen in Figure 47 when there was no noise added and the embedded and the extracted watermark are the same.

1) For the addition of the **salt and pepper and the Gaussian noise attacks**: the robustness of the proposed algorithm against the salt and the pepper noise is quite good as the extracted watermark is quite disfigured when compared to the embedded watermark as it is seen in Figure 49, and the correlation is shown in Figure 52. In both the salt and pepper and the Gaussian noise attacks the noise addition will affect each pixel in the watermarked frame. Which may not be visible at all times in the output video (Figure 50 and Figure 54). The embedded and the extracted watermarks after the addition of the gaussian noise is shown in Figure 53, the video obtained after the addition of the gaussian noise is in Figure 54 and the correlation of the embedded and the extracted watermark is shown in Figure 56, while the PSNR and the MSE values for the first 64 frames are shown in Figure 55.

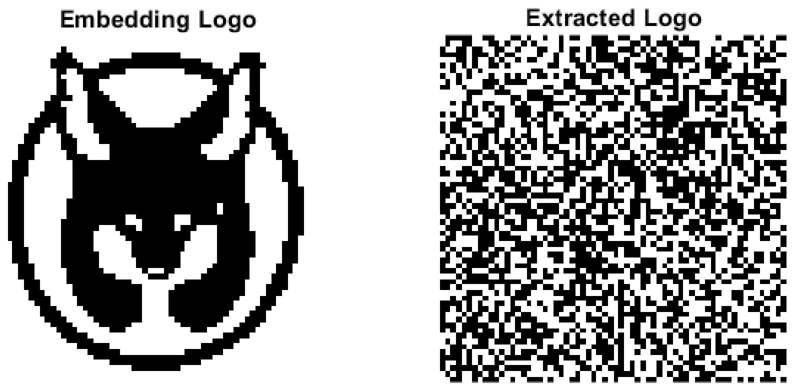


Figure 49 The Embedded and the Extracted Watermark after the addition of the Salt and Pepper noise

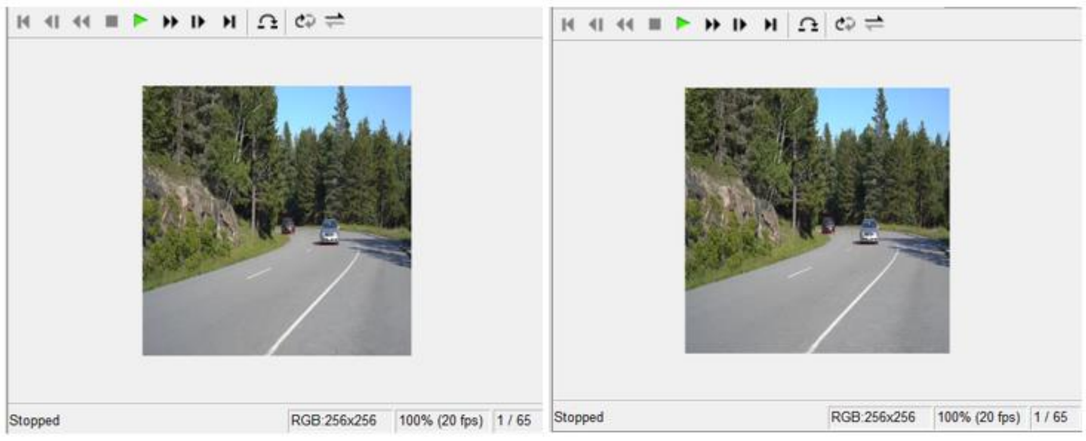
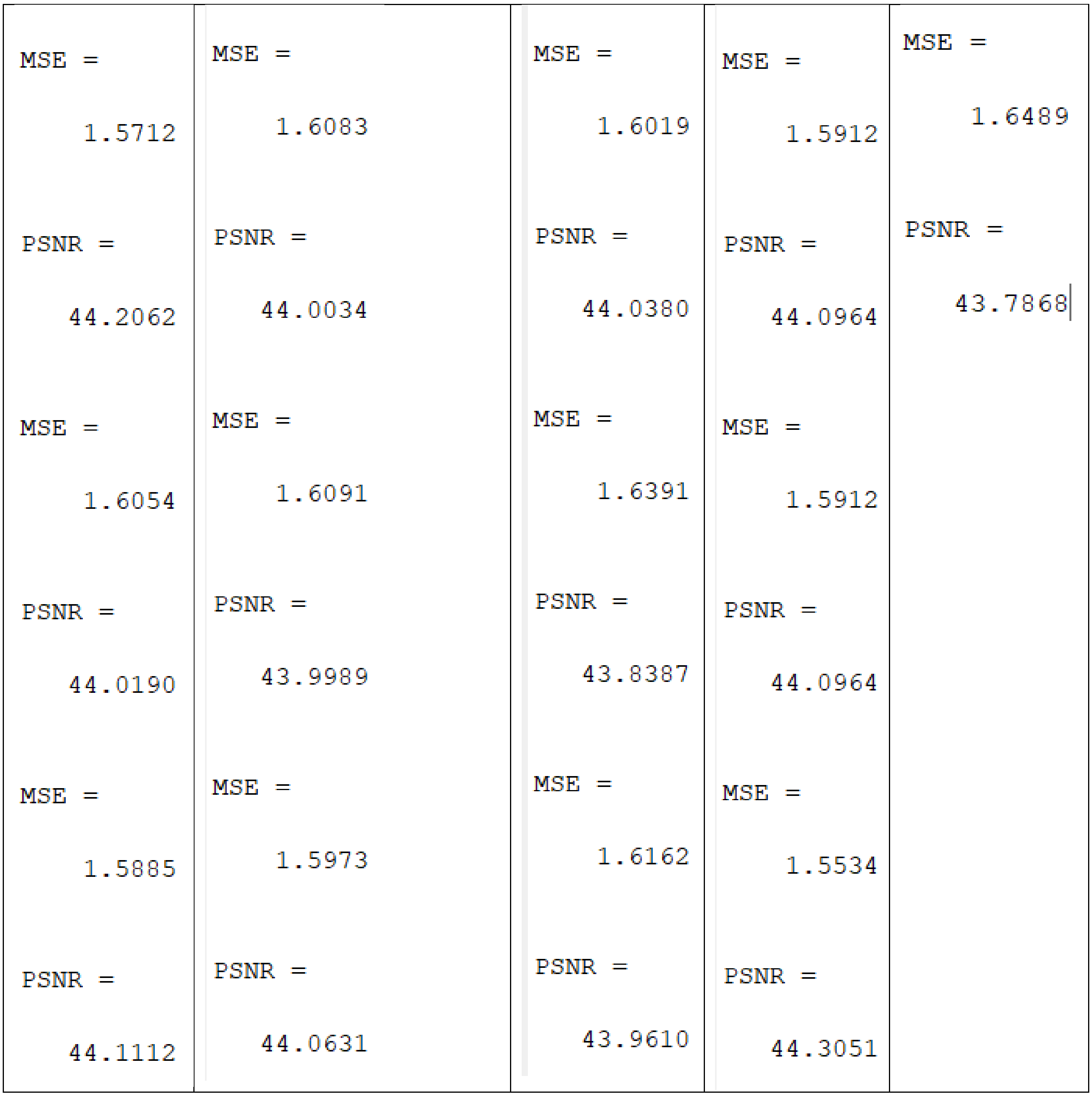


Figure 50 The Video before and after the Salt and Pepper noise attack

MSE = 1.5370	MSE = 1.5499	MSE = 1.5034	MSE = 1.5137	MSE = 1.4342	MSE = 1.5105
PSNR = 44.3971	PSNR = 44.3249	PSNR = 44.5893	PSNR = 44.5301	PSNR = 44.9988	PSNR = 44.5483
MSE = 1.5370	MSE = 1.5253	MSE = 1.5001	MSE = 1.5568	MSE = 1.4349	MSE = 1.5032
PSNR = 44.3971	PSNR = 44.4636	PSNR = 44.6087	PSNR = 44.2861	PSNR = 44.9942	PSNR = 44.5906
MSE = 1.6136	MSE = 1.5140	MSE = 1.4688	MSE = 1.5790	MSE = 1.4284	MSE = 1.4873
PSNR = 43.9750	PSNR = 44.5280	PSNR = 44.7914	PSNR = 44.1629	PSNR = 45.0340	PSNR = 44.6830

MSE =	MSE =	MSE =	MSE =	MSE =	MSE =
1.5105	1.5869	1.6107	1.5576	1.4928	1.6047
PSNR =	PSNR =	PSNR =	PSNR =	PSNR =	PSNR =
44.5483	44.1196	43.9908	44.2818	44.6505	44.0232
MSE =	MSE =	MSE =	MSE =	MSE =	MSE =
1.5032	1.5944	1.5346	1.5586	1.5311	1.6047
PSNR =	PSNR =	PSNR =	PSNR =	PSNR =	PSNR =
44.5906	44.0788	44.4109	44.2763	44.4309	44.0232
MSE =	MSE =	MSE =	MSE =	MSE =	MSE =
1.4873	1.5886	1.5954	1.5038	1.5298	1.5865
PSNR =	PSNR =	PSNR =	PSNR =	PSNR =	PSNR =
44.6830	44.1104	44.0733	44.5870	44.4380	44.1222

MSE = 1.6189	MSE = 1.5753	MSE = 1.6585	MSE = 1.7307	MSE = 1.5807	MSE = 1.5777
PSNR = 43.9467	PSNR = 44.1836	PSNR = 43.7366	PSNR = 43.3662	PSNR = 44.1541	PSNR = 44.1705
MSE = 1.5727	MSE = 1.5753	MSE = 1.6762	MSE = 1.7300	MSE = 1.5397	MSE = 1.5714
PSNR = 44.1978	PSNR = 44.1837	PSNR = 43.6441	PSNR = 43.3700	PSNR = 44.3821	PSNR = 44.2049
MSE = 1.5132	MSE = 1.6139	MSE = 1.7301	MSE = 1.6357	MSE = 1.5525	MSE = 1.5478
PSNR = 44.5330	PSNR = 43.9731	PSNR = 43.3695	PSNR = 43.8568	PSNR = 44.3103	PSNR = 44.3366



```
MSE =

    1.5712


PSNR =

   44.2062


MSE =

    1.6054


PSNR =

   44.0190


MSE =

    1.5885


PSNR =

   44.1112
```

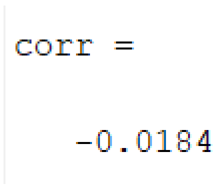
```
MSE =

    1.6083


PSNR =

   44.0034


MSE =

    1.6091


PSNR =

   43.9989


MSE =

    1.5973


PSNR =

   44.0631
```

```
MSE =

    1.6019


PSNR =

   44.0380


MSE =

    1.6391


PSNR =

   43.8387


MSE =

    1.6162


PSNR =

   43.9610
```

```
MSE =

    1.5912


PSNR =

   44.0964


MSE =

    1.5912


PSNR =

   44.0964


MSE =

    1.5534


PSNR =

   44.3051
```

```
MSE =

    1.6489


PSNR =

   43.7868
```

Figure 51 The PSNR and the MSE values for the first 64 Frames



```
corr =

   -0.0184
```

Figure 52 The Correlation between the Extracted Image and the Embedded Image with the addition of Salt and Pepper noise

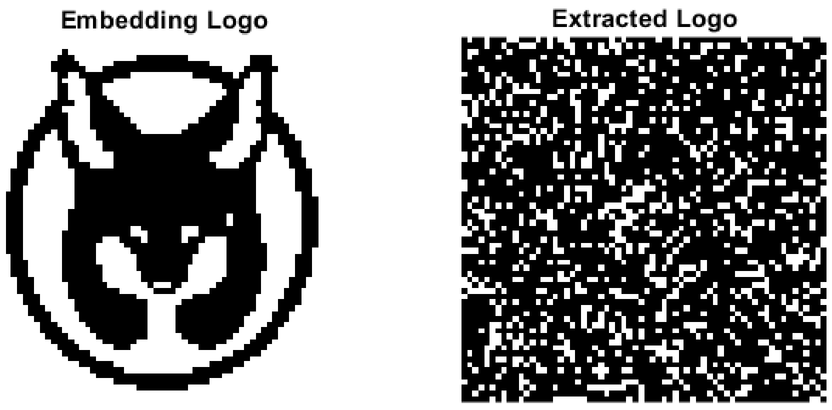


Figure 53 The Embedded and the Extracted Watermark after the addition of the Gaussian noise

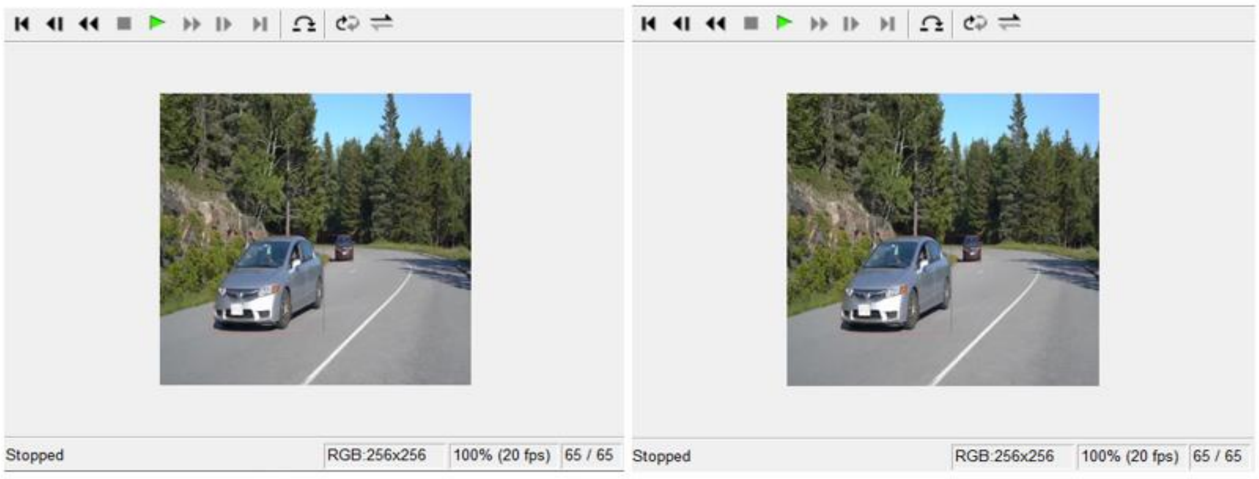


Figure 54 The Video before and after the Gaussian noise attack

MSE = 1.5370	PSNR = 44.3249	MSE = 1.5001	PSNR = 44.2861	MSE = 1.4284	PSNR = 44.6830
PSNR = 44.3971	MSE = 1.5253	PSNR = 44.6087	MSE = 1.5790	PSNR = 45.0340	MSE = 1.5869
MSE = 1.5370	PSNR = 44.4636	MSE = 1.4688	PSNR = 44.1629	MSE = 1.5105	PSNR = 44.1196
PSNR = 44.3971	MSE = 1.5140	PSNR = 44.7914	MSE = 1.4342	PSNR = 44.5483	MSE = 1.5944
MSE = 1.6136	PSNR = 44.5280	MSE = 1.5137	PSNR = 44.9988	MSE = 1.5032	PSNR = 44.0788
PSNR = 43.9750	MSE = 1.5034	PSNR = 44.5301	MSE = 1.4349	PSNR = 44.5906	MSE = 1.5886
MSE = 1.5499	PSNR = 44.5893	MSE = 1.5568	PSNR = 44.9942	MSE = 1.4873	PSNR = 44.1104

MSE = 1.6107	PSNR = 44.2818	MSE = 1.6047	PSNR = 44.1978	MSE = 1.6139	PSNR = 43.3695
PSNR = 43.9908	MSE = 1.5586	PSNR = 44.0232	MSE = 1.5132	PSNR = 43.9731	MSE = 1.7307
MSE = 1.5346	PSNR = 44.2763	MSE = 1.5865	PSNR = 44.5330	MSE = 1.6585	PSNR = 43.3662
PSNR = 44.4109	MSE = 1.5038	PSNR = 44.1222	MSE = 1.5753	PSNR = 43.7366	MSE = 1.7300
MSE = 1.5954	PSNR = 44.5870	MSE = 1.6189	PSNR = 44.1836	MSE = 1.6762	PSNR = 43.3700
PSNR = 44.0733	MSE = 1.4928	PSNR = 43.9467	MSE = 1.5753	PSNR = 43.6441	MSE = 1.6357
MSE = 1.5576	PSNR = 44.6505	MSE = 1.5727	PSNR = 44.1837	MSE = 1.7301	PSNR = 43.8568

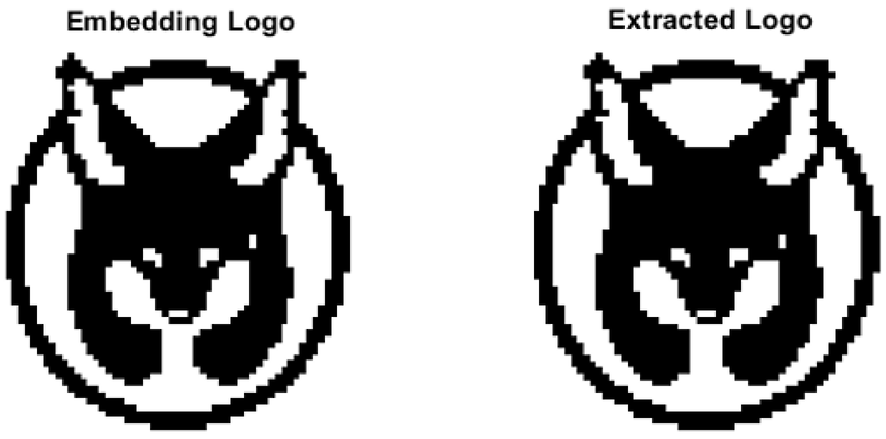
MSE = 1.5807	PSNR = 44.1705	MSE = 1.6054	MSE = 1.6019	PSNR = 44.0964
PSNR = 44.1541	MSE = 1.5714	PSNR = 44.0190	PSNR = 44.0380	MSE = 1.5912
MSE = 1.5397	PSNR = 44.2049	MSE = 1.5885	MSE = 1.6391	PSNR = 44.0964
PSNR = 44.3821	MSE = 1.5478	PSNR = 44.1112	PSNR = 43.8387	MSE = 1.5534
MSE = 1.5525	PSNR = 44.3366	MSE = 1.6083	MSE = 1.6162	PSNR = 44.3051
PSNR = 44.3103	MSE = 1.5712	PSNR = 44.0034	PSNR = 43.9610	MSE = 1.6489
MSE = 1.5777	PSNR = 44.2062	MSE = 1.6091	MSE = 1.5912	PSNR = 43.7868

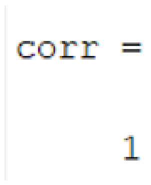
Figure 55 The PSNR and the MSE values for the first 64 Frames after the addition of the Gaussian noise

```
corr =

    0.0243
```

Figure 56 The Correlation between the Extracted Image and the Embedded Image with the addition Gaussian noise

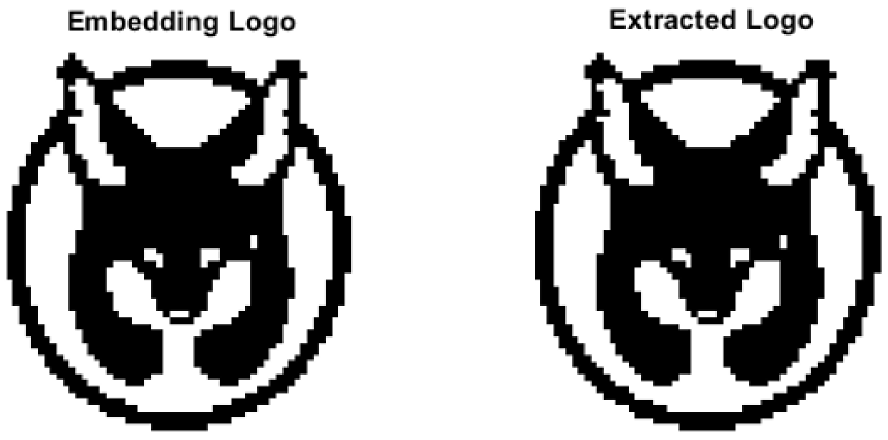
2) For the addition of **rotation attacks:** in this case, the watermarked frames are rotated by the angels 30^o, 90^o and 180^o. The proposed algorithm will show the robustness against these attacks. The embedded and the extracted watermark images as well as the correlation between the two watermarks are shown in Figure 57 (for 30^o), Figure 58 (for 90^o) and Figure 59 (for 180^o).

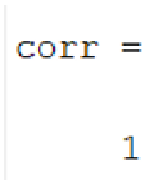




```
corr =

     1
```

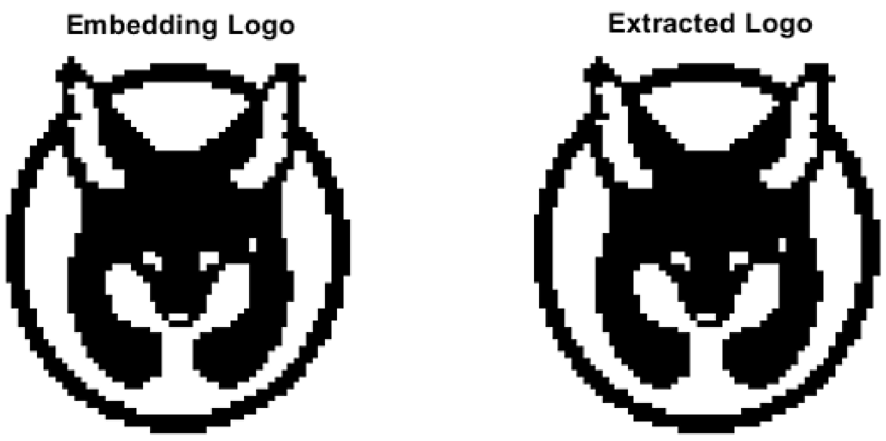
Figure 57 The Embedded and the Extracted Logo along with the Correlation (coefficient value 1) for a 30-degree rotation of the Frames

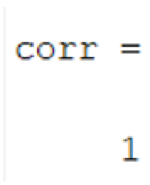




```
corr =

     1
```

Figure 58 The Embedded and the Extracted Logo along with the Correlation (coefficient value 1) for a 90-degree rotation of the Frames





```
corr =

     1
```

Figure 59 The Embedded and the Extracted Logo along with the Correlation (coefficient value almost 1) for a 180-degree rotation of the Frames

From Figures 57, 58, and 59 it can be seen that the watermarks are almost the same or are the same with high values of correlation with the coefficient is 1. Hence the diagonal in the S matrix was selected in this algorithm for the watermark embedding. Because of the eigenvalues present in the SCHUR decomposition that are very stable this algorithm helps increase the robustness.

The reason this is effective is that if the video or the respective frames are twisted or rotated it does not actually harm or tamper the video as much as noise does it just gets rotated by certain angles while the video or the frames themselves do not lose any data. Meaning after rotating the resulting output is the same video.

Chapter 8: Conclusion

In everyday life, there is various noise present while a video is being recorded or while taking clicking an image, these noise can come from sources such as the placement of the video recording or the image capturing camera, background irregularities, and disturbances such as ripples on water, birds flying in multiple directions. Due to these and many other factors detection of the moving objects get harder. For the identification and the detection of the moving objects, this report makes use of blob analysis more specifically background subtraction and morphological operations. With the aid of the object tracking method, a highly efficient and very flexible result is obtained when the moving objects are being detected. In addition to this, the background subtraction and the morphological operations in addition to the blob analysis is very efficient and very quick when it comes to tracking the moving objects. This is very helpful when these methods (tracking) are compared to the other algorithms available in the market. Once the objects (moving) are detected and tracked the video clip will be saved and then send to the Video (images) quality enhancement with highly reduced noise in the background.

The video quality enhancement unfortunately could only utilize the Image(s) deblurring algorithm as the other algorithms that were researched and considered required a different software (C language) for their respective uses, in this report the needed basic knowledge was introduced in the beginning, then the proposed algorithm applied to the video output obtained from the Detection and Tracking of the Moving Object(s) subsection and then the results and their before and after comparisons shown (Figure 40). The results did not show much of a change as the image sequences before and after the application of the algorithm are nearly the same, there was some difficulty in obtaining some blurred footage from various sources as the modern age video cameras generally record in High Definition. However, soon more experiments need to be done to study or prove the effectiveness of the algorithm proposed. A detailed study also must be done for other upcoming video enhancement methods and their applications regarding the repairing of the pixels if any damage did occur while the video clip was being saved in a given format. The proposed deblurring algorithm can be used more than once up to three more times to obtain a much better performance, keeping in mind this algorithm requires a lot of processing memory so a device with higher RAM (Random Access Memory) will be beneficial for double or triple deblurring performances. After this stage, the video generated after the individual frames are deblurred (or their contrast-enhanced) will be processed by the watermarking algorithm.

The report then introduces the video watermarking algorithm more specifically a blind and robust watermarking algorithm for video formats. The proposed algorithm makes use of the SCHUR decomposition and the Discrete Wavelet Transform. From the results observed from Figure 44 to Figure 56, it can be observed that the extracted watermark indeed does protect against noise attacks since there is no distortion present in the extracted watermark present while processing the video obtained from the Video Quality Enhancement (deblurring) sub-section which had no noise. And when gaussian and Salt and Pepper noise was added the distorted extracted watermark can be seen very easily concluding some tampering was done. Also, the proposed algorithm makes use of the SCHUR decomposition which because of the small number of calculations and computations occurring ensures the improvement of the efficiency of this algorithm when compared to the other decomposition algorithms.

Lastly, this report was made and researched by combining three major functions of Image and Video processing into one bulk program. Ironically so far very few researchers have fiddled with the idea of combining these 3 major functions presumably due to its high processing needs.

Future Scope and Applications

Unfortunately, MATLAB does not support two lines of code them being Hasframe and Readframe. These two lines of code had MATLAB continued using would be very helpful in the following ways:

1) Instead of detecting and tracking objects on a still background, these two lines could be used in the YOLO (you look only once) version 1 to 3 algorithms for detecting cars, pedestrians, traffic lights, road signs, speed limits on a moving background (preferably on live dashcam footage). However, this is still possible if the codes are exported to Python or Open CV software platforms that use a different set of codes but utilize the same algorithm and give the same results.

2) MATLAB would be able to create and read from a convolution neural network (R-CNN) provided the software has access to the specific external hardware and GPU (Graphics Processing Unit). Reading data and formats from a CNN is still a challenging task which is quite possible even without the two lines of code mentioned above. The only requirement would be a high-end GPU to create the CNN and a code generator or another NVIDIA GPU to obtain data from the CNN and process it. The main advantage here would be the software would scan everything present in the video (irrespective of background motion) and compare it to the images obtained from CNN and identify those objects only which are of interest. It can be used to track only pedestrians from a dashcam video, or only cars for automated driving, or only traffic lights and their respective meanings.

The YOLO algorithm is the future of this report as it has proper use in automated cars, self-driving vehicles, security devices (mobile and stationary). If combined with a low-light enhancement algorithm the vehicles (civilian and military) would be able to detect movements in the dark. Increased protection offered by security cameras. With the help of CNN, the police will be able to detect and track an object solely from a cluttered environment.

Another future possibility would be if the algorithms related to face detection and tracking utilizing KLT and RANSAC, could be included within the object detection and tracking algorithm so the CCTV in addition to capturing moving cars, trucks, busses, cycles will also detect and capture the image of the person(s) present in the vehicle of course assuming the footages are of high definition. Or perhaps, If the text extraction by OCR algorithms can be included, with this algorithm the software will be able to extract texts such as characters, numbers, alphabets, symbols on a level background (traffic signs, number plates, posters, banners, road signs), to help retrieve some details such as license plate numbers (most important application), diversion routes, road names, road directions on banners.

With some minor adjustments, even facial data and expressions can be recorded and stored as data which will be helpful in student attendance, identifying the employees in an industry/organization or the police can connect it to their base and increase the range of identifying and capturing criminals (the accuracy will be completely dependent on the Convolution neural network).

This report however was made keeping in mind about the security cameras in casual shops and the amount of protection (less or high) they provide to the owners, this will help in detecting if there was any robbery conducted by alerting the owner(s) through mail along with enhanced images, with each image (frame of a video) watermarked for protection against different forms of tampering and attacks. It can be used in any place, regardless of the amount of money spent in terms of security.

Works Cited

[1] Aggarwal, R. M. (2010). A Comprehensive Review of Image Enhancement Techniques. *JOURNAL OF COMPUTING, VOLUME 2, ISSUE 3, ISSN 2151-9617*.

[2] Al-Khatib, T. A.-H. (2008). A Robust Video Watermarking Algorithm. *Journal of Computer Science Volume 4, No. 11*, 910-915. Retrieved from Al-Khatib, T., Al-Haj, A., Rajab, L. & Mohammed, H. (2008). A Robust Video Watermarking Algorithm. Journal of Computer Science, 4(11), 910-915.: https://doi.org/10.3844/jcssp.2008.910.915

[3] *Al-Khatib, T., Al-Haj, A., Rajab, L. & Mohammed, H. (2008). A Robust Video Watermarking Algorithm. Journal of Computer Science, 4(11), 910-915. https://doi.org/10.3844/jcssp.2008.910.915*. (n.d.).

[4] B.Chandra Mohan, K. S. (2011). A Comparative performance evaluation of SVD and Schur Decompositions for Image Watermarking. *IJCA Proceedings on International Conference on VLSI, Communications and Instrumentation (ICVCI) (14)*, 25-29.

[5] Baisa L Gunjal, R. M. (2010). An overview of transform domain robust digital image watermarking algorithms. *Journal of Emerging Trends in Computing and Information Sciences, Volume 2, Issue 1*, 37-42. Retrieved from Gunjal, B.L. and Manthalkar, R.R. An overview of Transform Domain Robust Digital Image Watermarking Algorithms –Baisa L. Gunjal, R. R. Manthalkar — Journal of Emerging trends in Computing and Information Science, (2) 2010, pp 37- 42: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.302.6848

[6] Berge, J. M. (1993). *Least Squares Optimization in Multivariate Analysis.* Leiden: DSWO Press.

[7] BOVIK, A. (2000). *Handbook of Image and Video Processing, Academic Press Series in Communications, Networking, and Multimedia.* San Diego, USA: Academic Press .

[8] C. Stauffer, W. G. (1999). Adaptive background mixture models for real-time tracking. *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)* (pp. 246-252). Fort Collins, CO, USA, USA: IEEE.

[9] C.B. Atkins, C. B. (7-10 Oct. 2001). Optimal image scaling using pixel classification. *Proceedings 2001 International Conference on Image Processing (Cat. No.01CH37205)* (pp. 864-867). Thessaloniki, Greece, Greece: IEEE.

[10]Charles F. Van Loan, G. H. (2012). *Matrix Computations, 4th edition.* Baltimore: The Johns Hopkins University Press.

[11]D. Kundur, D. H. (1998, May 15). Digital watermarking using multiresolution wavelet decomposition. *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98 (Cat. No.98CH36181)*, 2969-2972.

[12]Dipali Shahare, R. S. (2014). Moving Object Detection with Fixed Camera and Moving Camera for Automated Video Analysis. *International Journal of Computer Applications Technology and Research, Volume 3-Issue 5*, 277 - 283.

[13]Efford, N. (2000). *Digital Image Processing: A Practical Introduction using Java.* United States: Pearson Education. Retrieved from Auckland.

[14]Efford, N. (2000). *Morphological Image Processing*. Retrieved from aukland: https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic4.htm

[15]GAVIN, B. (2018, September 12). *What Is an AVI File (and How Do I Open One)?* Retrieved from How-to Geek: https://www.howtogeek.com/365627/what-is-an-avi-file-and-how-do-i-open-one/#:~:text=An%20AVI%20file%20uses%20less,GB%20per%20minute%20of%20video.

[16]H. Hu, G. d. (10-14 Jan. 2007). Simultaneous Coding Artifact Reduction and Sharpness Enhancement. *2007 Digest of Technical Papers International Conference on Consumer Electronics.* Las Vegas, NV, USA: IEEE.

[17]Hario Baskoro Basoeki, A. D. (2016). Improving sperms detection and counting using single Gaussian background subtraction. *https://ieeexplore.ieee.org/servlet/opac?punumber=7859013*.

[18]Hassen Seddik, M. S. (2009). A new blind image watermarking method based on shur transformation. *2009 35th Annual Conference of IEEE Industrial Electronics*, 1967-1972.

[19]Hong, W. a. (2006). Robust Digital Watermarking Scheme for Copy Right Protection. *IEEE Transactions on Signal Processing, Issue 12*, 1-8.

[20]Hussein, J. A. (Jan 2010, January 20). Retrieved from Hussein, J.A. (2010) Spatial Domain Watermarking Scheme for Colored Images Based on Log-Average Luminance. CoRR, abs/1001.3496.: https://arxiv.org/abs/1001.3496

[21]*IEEE Communications Magazine*. (n.d.). Retrieved from Voloshynovskyy, S., Pereira, S., Pun, T., Eggers, J.J. & Su, J.K. 2001, "Attacks on Digital Watermarks : Classification, Estimation-based Attacks and Benchmarks", IEEE Communications Magazine, vol. 39, no. 8, pp. 118-127.

[22]K Meenakshi, C. S. (2014). a fast and robust hybrid watermarking scheme based on Schur and Svd transform. *International Journal of Research in Engineering and Technology, Cover 3, Prescription 4*, 7-11.

[23]L.Shao, I. (January 2007). Content adaptive coding artifact reduction for decompressed video and Images. *proceddings of International Conference on Consumer Electronics (ICCE '07).* Las vegas, Nev, USA: IEEE.

[24]Lama Rajab, T. A.-K.-H. (2008). Hybrid DWT-SVD video watermarking. *2008 International Conference on Innovations in Information Technology*, 588-592.

[25]Lama Rajab, T. A.-K.-H. (2009/8). Video watermarking algorithms using the SVD transform. *European Journal of Scientific Research, Volume 30, No. 3*, 389-401.

[26]Li, X., & Orchard, M. (2001). New edge-directed interpolation. *IEEE Transactions on Image Processing, Volume: 10, Issue: 10*, 1521-1527.

[27]Ling Shao, H. H. (16 Sept.-19 Oct. 2007). Coding Artifacts Robust Resolution Up-conversion. *2007 IEEE International Conference on Image Processing.* San Antonio, TX, USA: IEEE.

[28]Ling Shao, H. Z. (2008). An Overview and Performance Evaluation of Classification-Based Least Squares Trained Filters. *IEEE Transactions on Image Processing, Volume: 17, Issue: 10*, 1772 - 1782.

[29]Liyuan Li, W. H. (2003). Foreground Object Detection from Videos ContainingComplex Background. *Proceedings of the Eleventh ACM International Conference on Multimedia, Berkeley* (pp. 2-10). CA, USA: Association for Computing Machinery, NewYork, USA.

[30]M.C. Hernandez, M. M. (2005). Analysis of a DFT-based watermarking algorithm. *2nd International Conference on Electrical and Electronics Engineering*, 44-47.

[31]Mohammad, A. A. (2012). A new digital image watermarking scheme based on Schur decomposition. *Multimedia Tools and Applications volume 59*, 851–883.

[32]Mondal, M. a. (Jan 2012). Spatial Domain Robust Watermarking Scheme forColor Image. *International Jounral of Advanced Computer Science, Vol. 2, No. 1*, 24-27.

[33]Owens, R. (1997, October 29). *Mathematical morphology*. Retrieved from HomePages: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT3/node3.html

[34]P. Thévenaz, T. B. (2000). Image Interpolation and Resampling. In I. Bankman, *Handbook of Medical Imaging, Processing, and Analysis* (pp. 393-420). San Diego CA, USA: Academic Press.

[35]Panahi, N. A. (2013). A New Colour Image Watermarking Scheme Using Cellular Automata Transform and Schur Decomposition. *Proceedings of the 21st Iranian Conference on Electrical Engineering (ICEE)*, 1-5.

[36]Philippe Thevenaz, T. B. (2000). Image Interpolation and Resampling (2000). *HANDBOOK OF MEDICAL IMAGING, PROCESSING, AND ANALYSIS*, 393-420.

[37]Piccardi, M. (2004). Background subtraction techniques: a review. *2004 IEEE International Conference on Systems, Man, and Cybernetics (IEEE Cat. No.04CH37583)* (pp. 3099-3103). The Hague, Netherlands: IEEE.

[38]PRATT, W. K. (2007). *Digital Image Processing, Fourth Edition.* Los Santos, California: John Wiley & Sons, Inc.

[39]*Principles of research ethics*. (n.d.). Retrieved from Laerd Dissertation: http://dissertation.laerd.com/principles-of-research-ethics.php

[40]Rafael C. Gonzalez, R. E. (2002). Digital Image Processing. 523-527.

[41]Rafael C. Gonzalez, R. E. (2008). *Digital Image processing Third Edition.* Upper Saddle River, New Jersey: Pearson Education, Inc.

[42]Rahman, M. M. (June 2013). A DWT, DCT, AND SVD BASED WATERMARKING TECHNIQUE TO PROTECT THE IMAGE PIRACY. *International Journal of Managing Public Sector Information and Communication Technologies (IJMPICT), vol 4, No. 2*, 21-31.

[43]Rosa Lancini, F. M. (2002, November 07). Retrieved from Lancini, R., Francesco Mapelli, and Stefano Tubaro. "A robust video watermarking technique in the spatial domain." International Symposium on VIPromCom Video/Image Processing and Multimedia Communications (2002): 251-256. 10.1109/VIPROM.2002.1026664: https://ieeexplore.ieee.org/document/1026664

[44]Sadik Ali M. Al-Taweel, P. S. (2009). Digital Video Watermarking in the Discrete Cosine Transform Domain. *Journal of Computer Science, volume 5 no 8*, 536-543.

[45]Salvi, G. (2012). An Automated Vehicle Counting System Based on Blob Analysis for Traffic Surveillance. *International conference, Image processing, computer vision, & pattern recognition; IPCV 2012; 2012; Las Vegas, NV* (pp. 397-402). Las Vegas, Nevada, United States: CSREA Press.

[46]Salwa A.K Mostafa, A. S. (2009). Video Watermarking Scheme Based on Principal. *IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.8, August 2009*, 45-52.

[47]Sathik, M. a. (2012). A Novel DWT Based Invisible Watermarking Technique for Digital Images. *International Arab Journal of e-Technology, Vol. 2*, 167-172.

[48]Satoshi Suzuki, K. b. (1985). Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing, Volume 30, Issue 1*, 32-46.

[49]Shan, K. K. (January 2015). Video Watermarking Using DCT and DWT: A Comparison. *European Journal of Advances in Engineering and Technology, 2015, 2(6)*, 83-87.

[50]Shao, L. (2008). Simultaneous coding artifact reduction and sharpness enhancement for block-based compressed images and videos. *Signal Processing: Image Communication, Volume 23, Issue 6*, 463-470.

[51]Shuiwang Ji, W. X. (2013, June). 3D Convolutional Neural Networks for Human Action Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence Volume: 35, Issue: 1*, 221 - 231. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Convolutional_neural_network

[52]Swamy, B. C. (2010). On the use of Schur Decomposition for Copyright Protection of Digital Images. *International Journal of Computer and Electrical Engineering, Volume 2, Number 4*, 781-787.

[53]Tao Jia, n.-L. S.-Y. (2008). Moving object detection based on blob analysis. *IEEE International Conference on Automation and Logistics*.

[54]Tetsujiro Kondo, K. K. (1995). *US Patent No. US5444487A.*

[55]Tetsujiro Kondo, Y. T. (2002). *United States Patent No. US6483546B1.*

[56]Umme Sara, M. A. (2019). Image Quality Assessment through FSIM, SSIM, MSE, and PSNR—A Comparative Study. *Journal of Computer and Communications, Vol.7 No.3, March 2019*, 2327-5227.

[57]Voloshynovskyy, S. P. (2001). Attacks on Digital Watermarks : Classification, Estimation-based Attacks and Benchmarks. *IEEE Communications Magazine. 2001, vol. 39, no. 8*, 118-127. Retrieved from VOLOSHYNOVSKYY, Svyatoslav et al. Attacks on Digital Watermarks : Classification, Estimation-based Attacks and Benchmarks. In: IEEE Communications Magazine, 2001, vol. 39, n° 8, p. 118-127.: https://archive-ouverte.unige.ch/unige:47520

[58]Xiang-Gen Xia, C. B. (1997, October 26-29). A multiresolution watermark for digital images. *Proceedings of International Conference on Image Processing*, 548-551.

[59]Yovits, M. C. (1980). *Advances in Computers, Vol. 18.* Toronto: Academic Press.

Appendix A

This section will contain all the extra results of this report,

1) **Frame(s) before and after Background Subtraction and Image(s) Morphology**

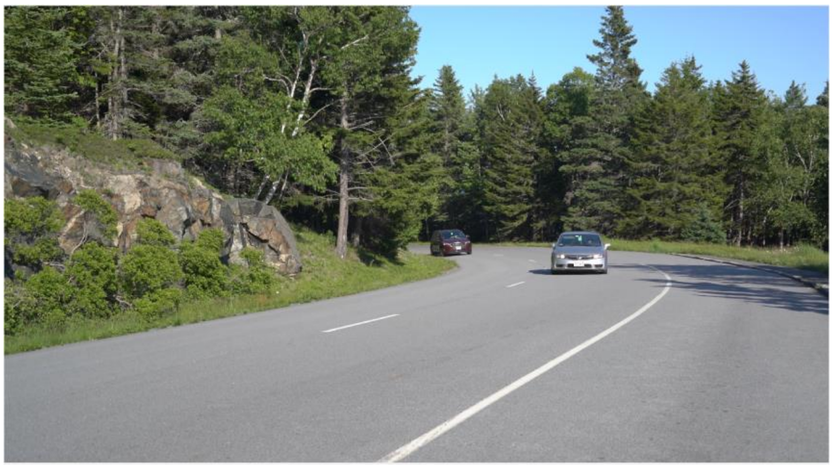


Figure 60 Input Frame of the Other Sample Video 2

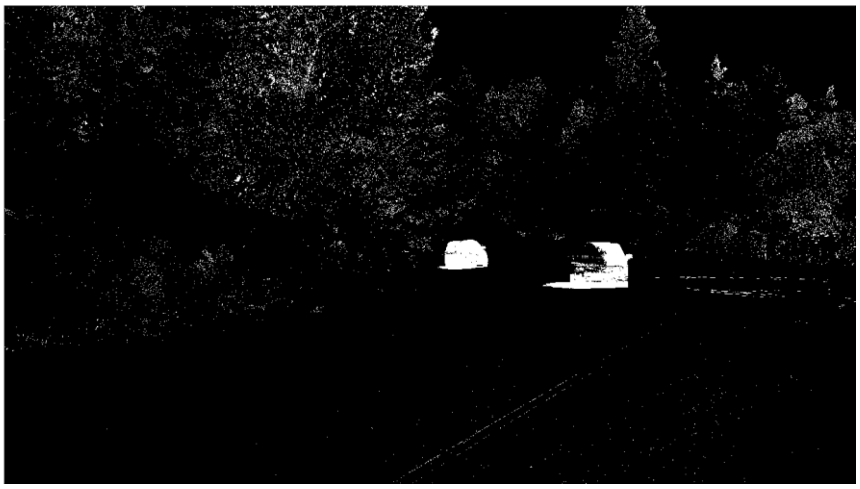


Figure 61 The resulting Frame of the Sample Video 2 after Background Subtraction with some visible noise

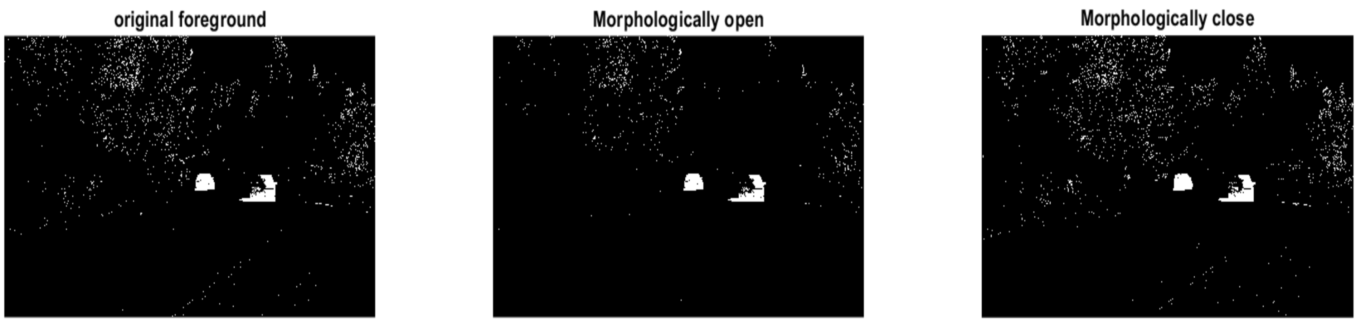


Figure 62 The Original Foreground from (left), the Open Morphological Operation (middle), the Close Morphological Operation (right) of the Sample Video 2

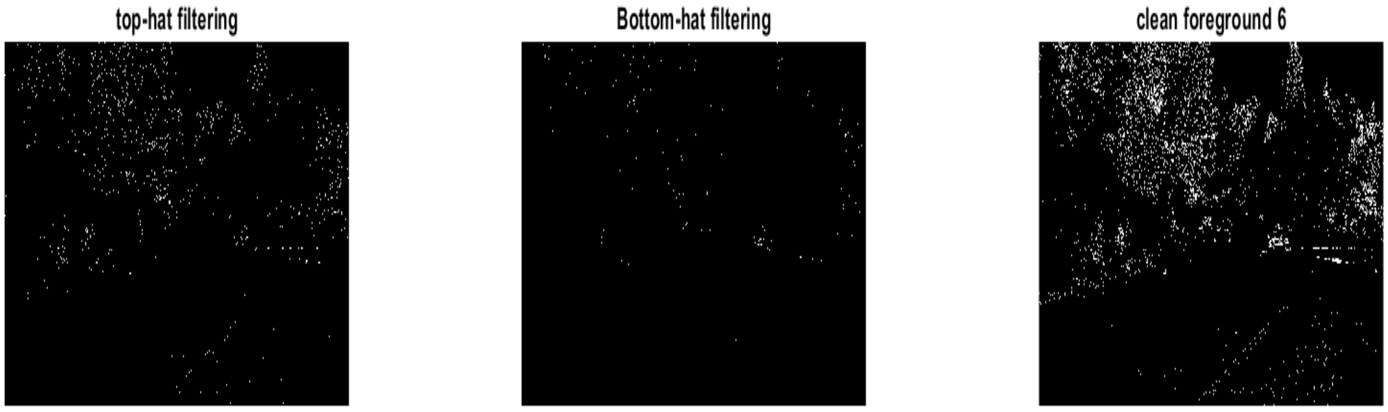


Figure 63 The Top-hat Morphological Operation (left), the Bottom-hat Morphological Operation (middle), the Morphological Operation obtained by subtracting Dilate and Erode (from Figure 61) of the Sample Video 2

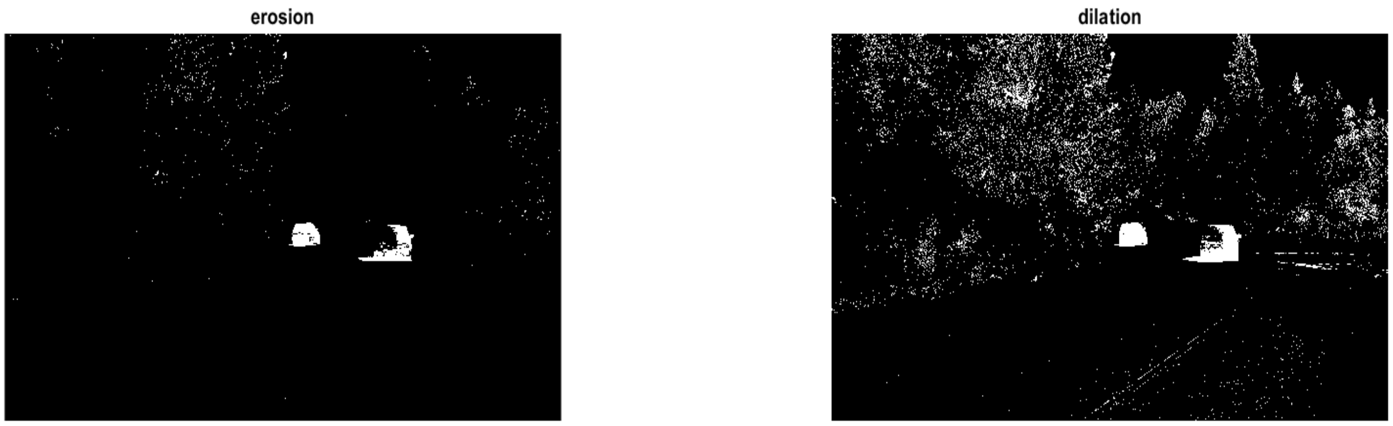


Figure 64 The Erode Morphological Operation (left), the Dilate Morphological Operation (right) of the Sample Video 2

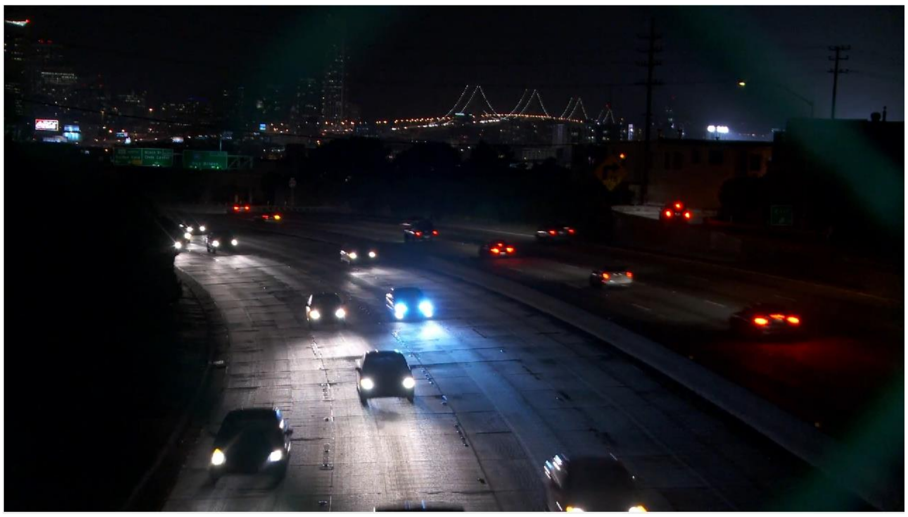


Figure 65 Input Frame of the Other Sample Video 3

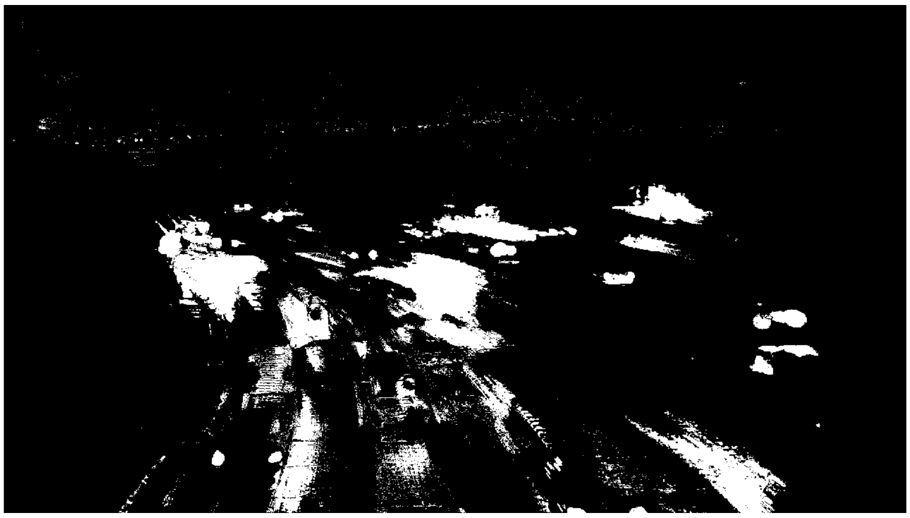


Figure 66 The resulting Frame of the Sample Video 3 after Background Subtraction with some visible noise

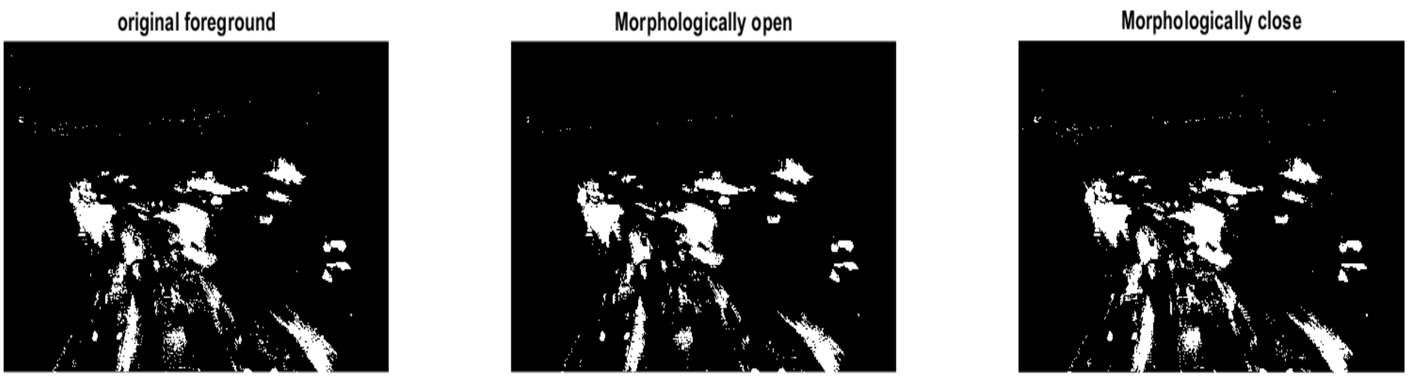


Figure 67 The Original Foreground from (left), the Open Morphological Operation (middle), the Close Morphological Operation (right) of the Sample Video 3

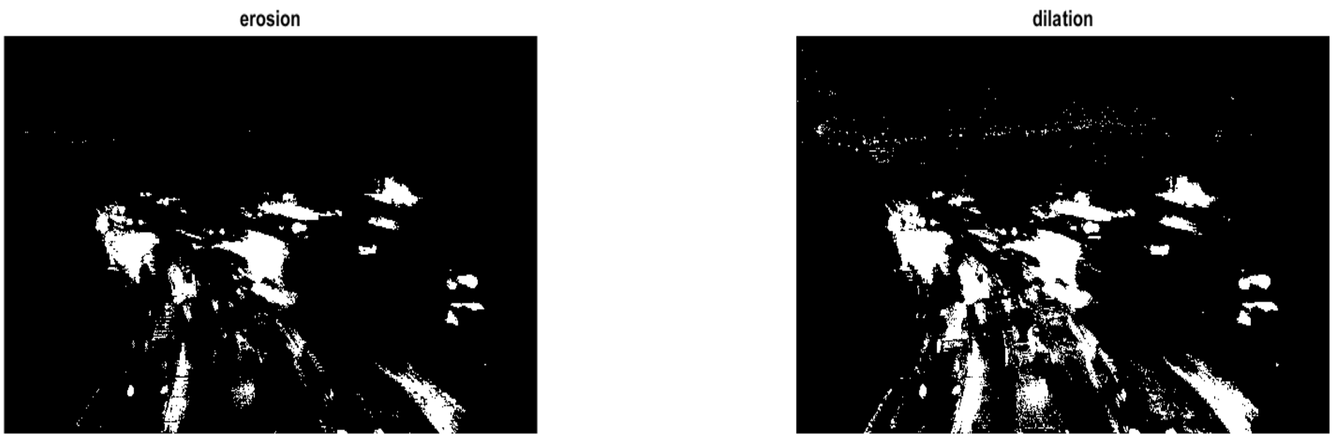


Figure 68 The Erode Morphological Operation (left), the Dilate Morphological Operation (right) of the Sample Video 3

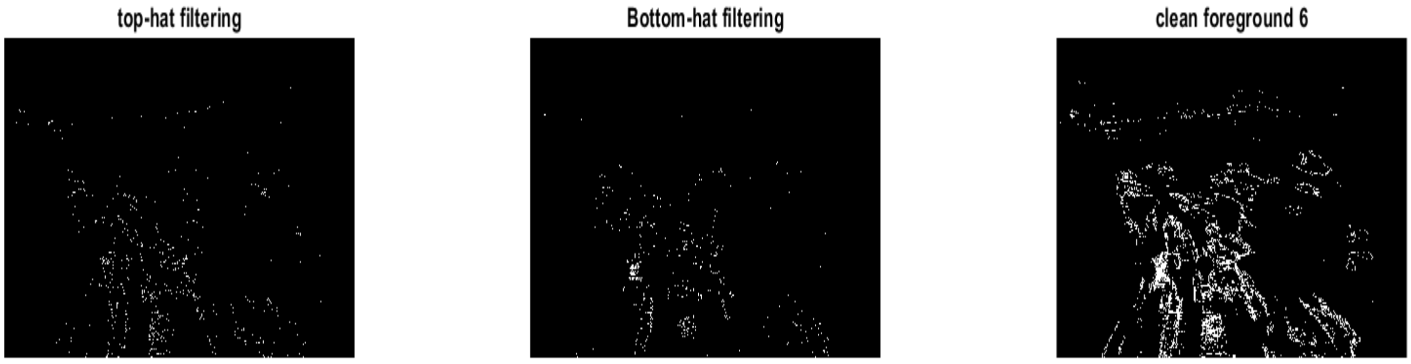


Figure 69 The Top-hat Morphological Operation (left), the Bottom-hat Morphological Operation (middle), the Morphological Operation obtained by subtracting Dilate and Erode of the Sample Video 3

2) Image(s) Enhancement

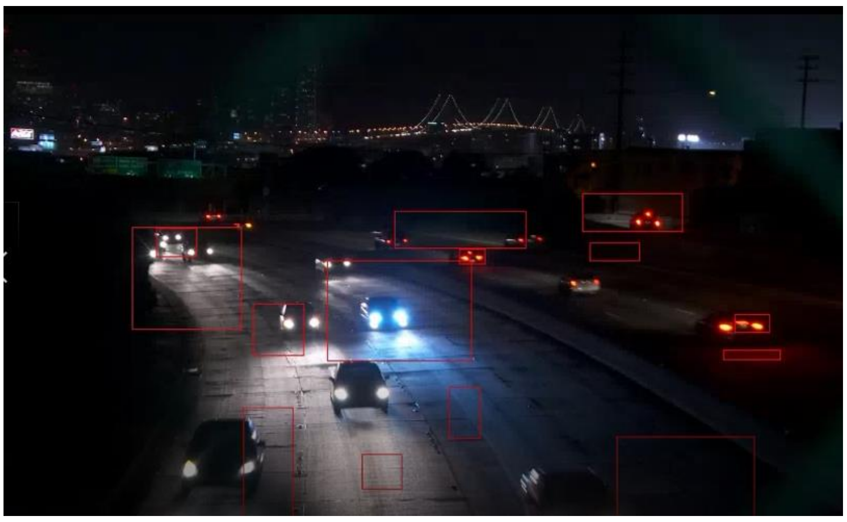


Figure 70 The Frame of the Output Video before the Image(s) Enhancement Algorithm is applied to Sample Video 3

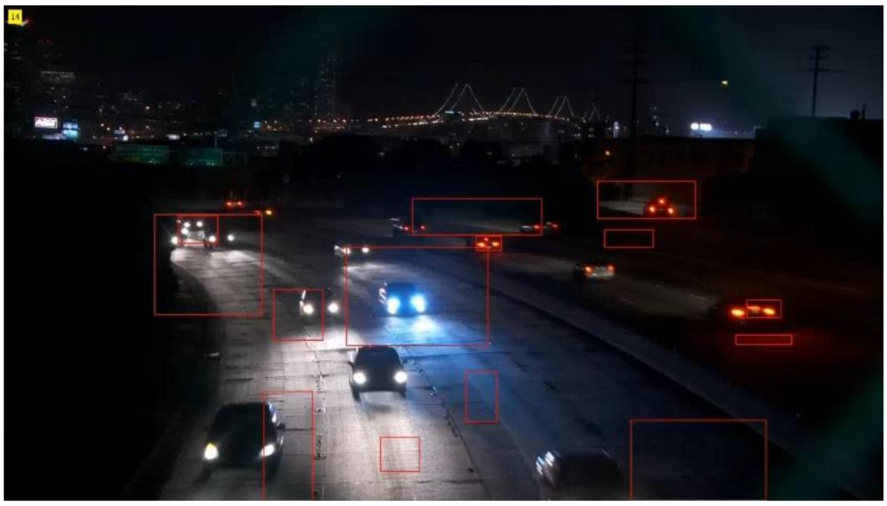


Figure 71 The same frame captured in Figure 67 after the application of the image(s) enhanced algorithm for sample video 3

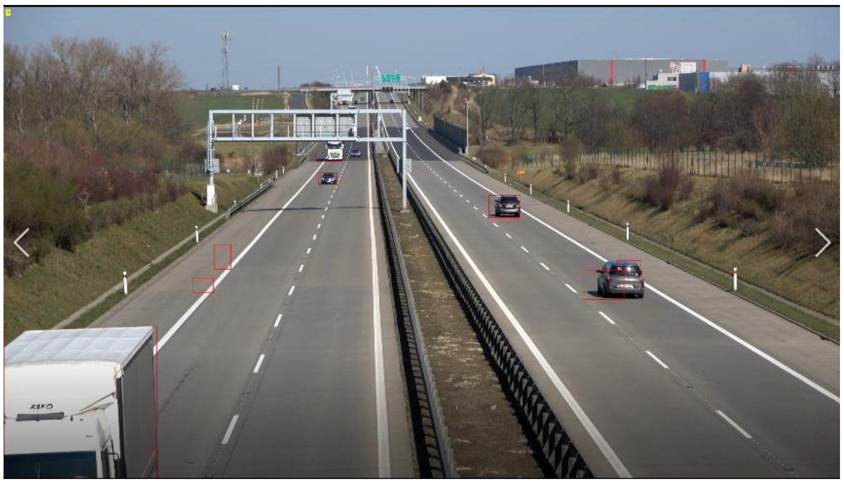


Figure 72 The Frame of the Output Video before the Image(s) Enhancement Algorithm is applied to Sample Video 1

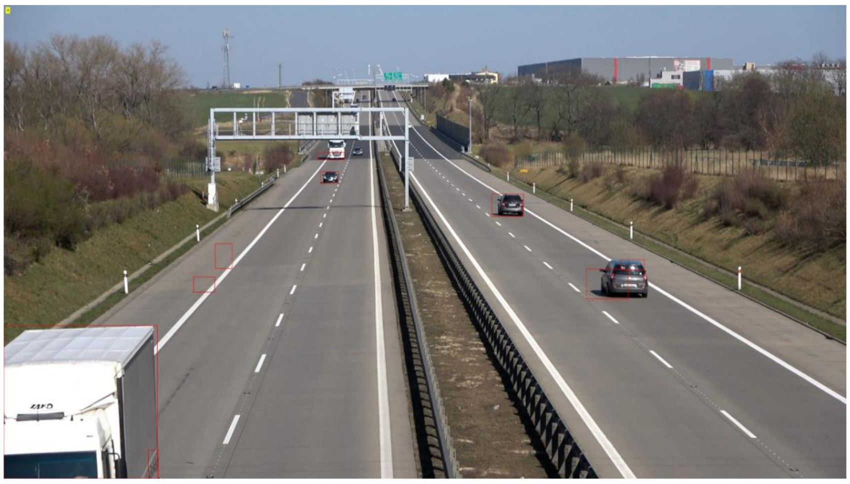


Figure 73 The same Frame captured in Figure 67 after the application of the Image(s) Enhanced Algorithm for Sample Video 1

3) Watermark and the addition of attacks for comparison

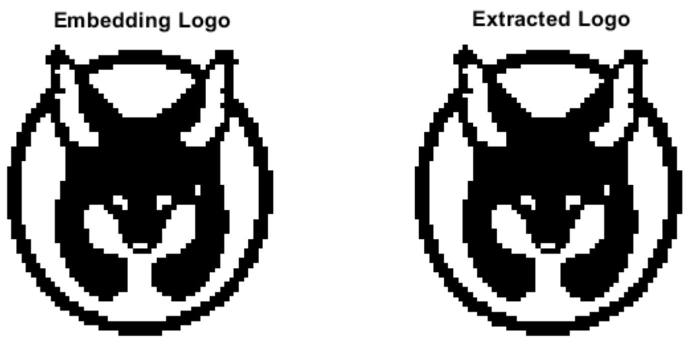


Figure 74 The Embedded and the Extracted Watermark from Sample Video 1 without any noise or tampering

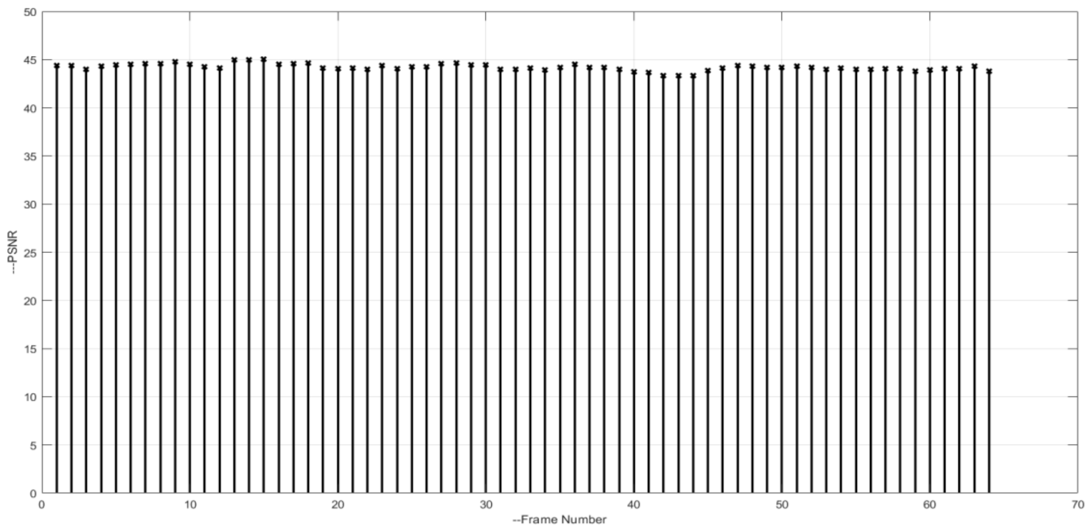


Figure 75 The PSNR values for the first 64 Watermarked Frames of Sample Video 1

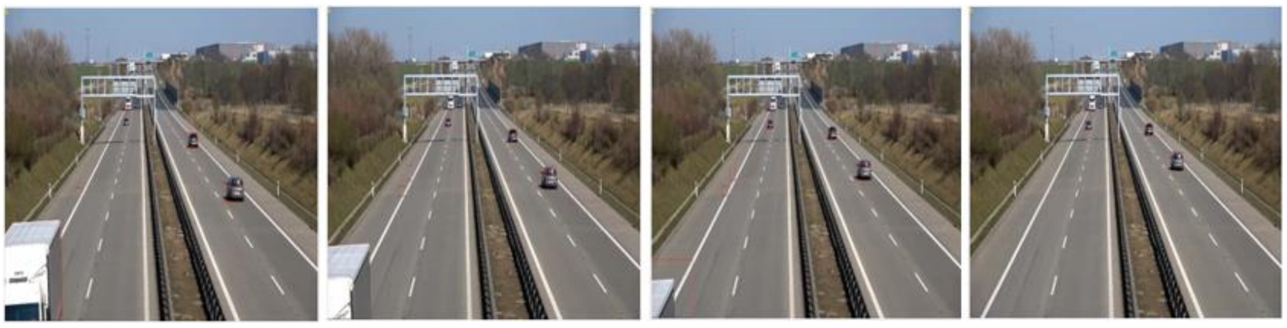


Figure 76 Frames captured from Sample Video 1 before Embedding the Watermark

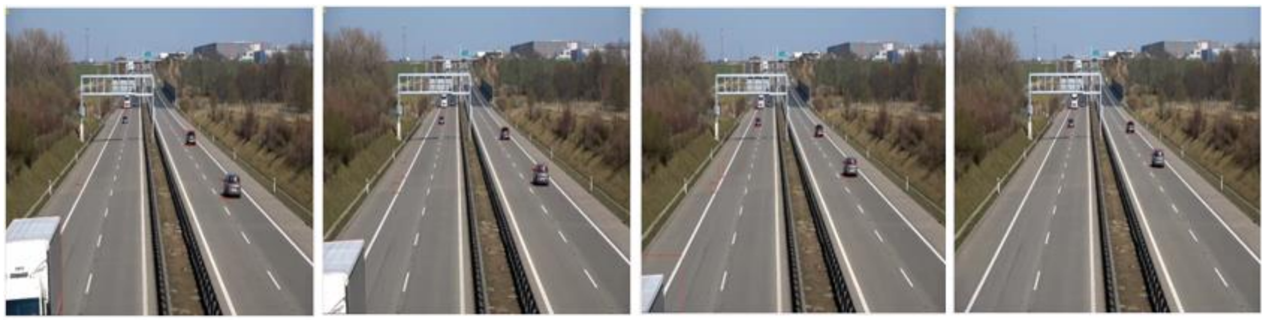


Figure 77 Frames captured from Sample Video 1 after Embedding the Watermark

```
corr =

     1
```

Figure 78 The Correlation between the Extracted and the Embedded Watermark without any noise for Sample Video 1

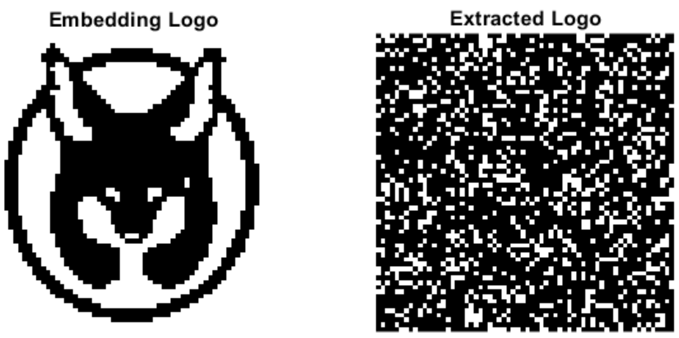
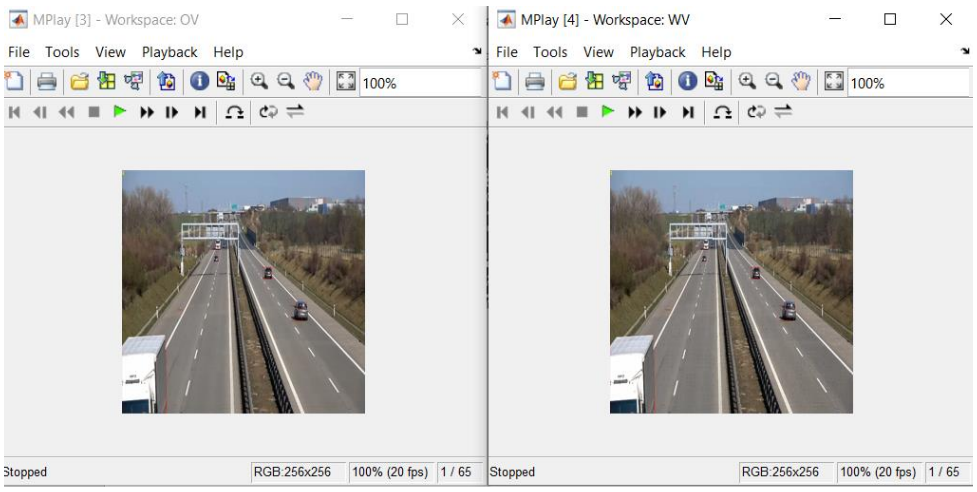


Figure 79 The Embedded and the Extracted Watermark after the addition of the Gaussian noise for Sample Video 1



MPlay [3] - Workspace: OV
File Tools View Playback Help
100%
Stopped
RGB:256x256
100% (20 fps)
1 / 65
MPlay [4] - Workspace: WV
File Tools View Playback Help
100%
Stopped
RGB:256x256
100% (20 fps)
1 / 65

Figure 80 The Video before and after the Gaussian noise attack for Sample Video 1

```
corr =

    0.0243
```

Figure 81 The Correlation coefficient of the Embedded and the Extracted Watermark after the addition of the Gaussian noise for Sample Video 1

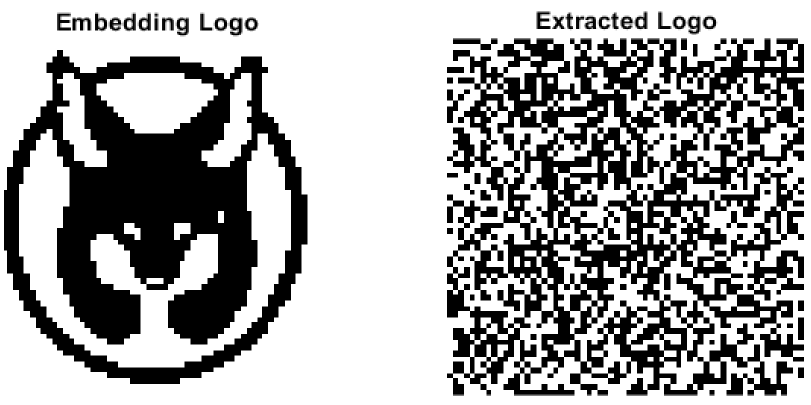


Figure 82 The Embedded and the Extracted Watermark after the addition of the Salt and Pepper noise in Sample Video 1

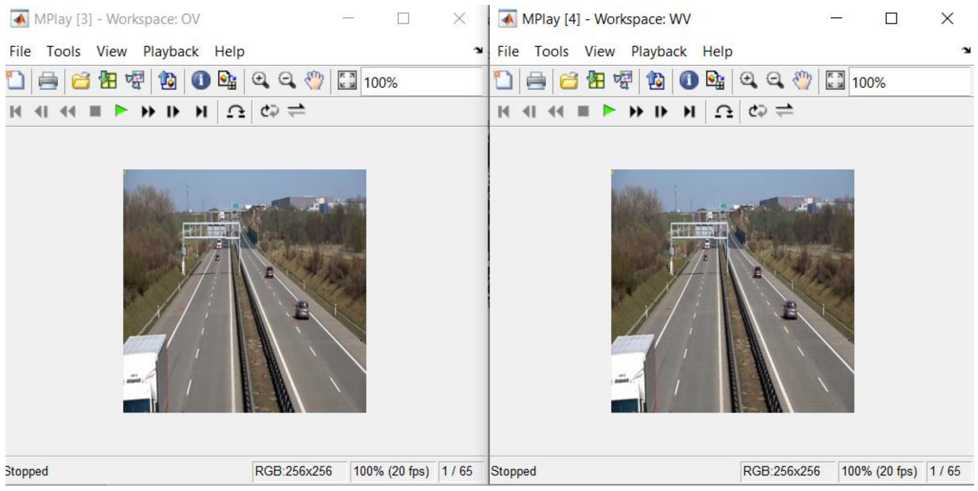
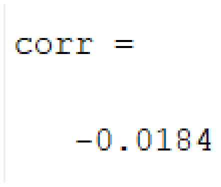

MPlay [3] - Workspace: OV
File Tools View Playback Help
100%
Stopped
RGB:256x256
100% (20 fps)
1 / 65
MPlay [4] - Workspace: WV
File Tools View Playback Help
100%
Stopped
RGB:256x256
100% (20 fps)
1 / 65

Figure 83 The Video before and after the Salt and Pepper noise attack for Sample Video 1



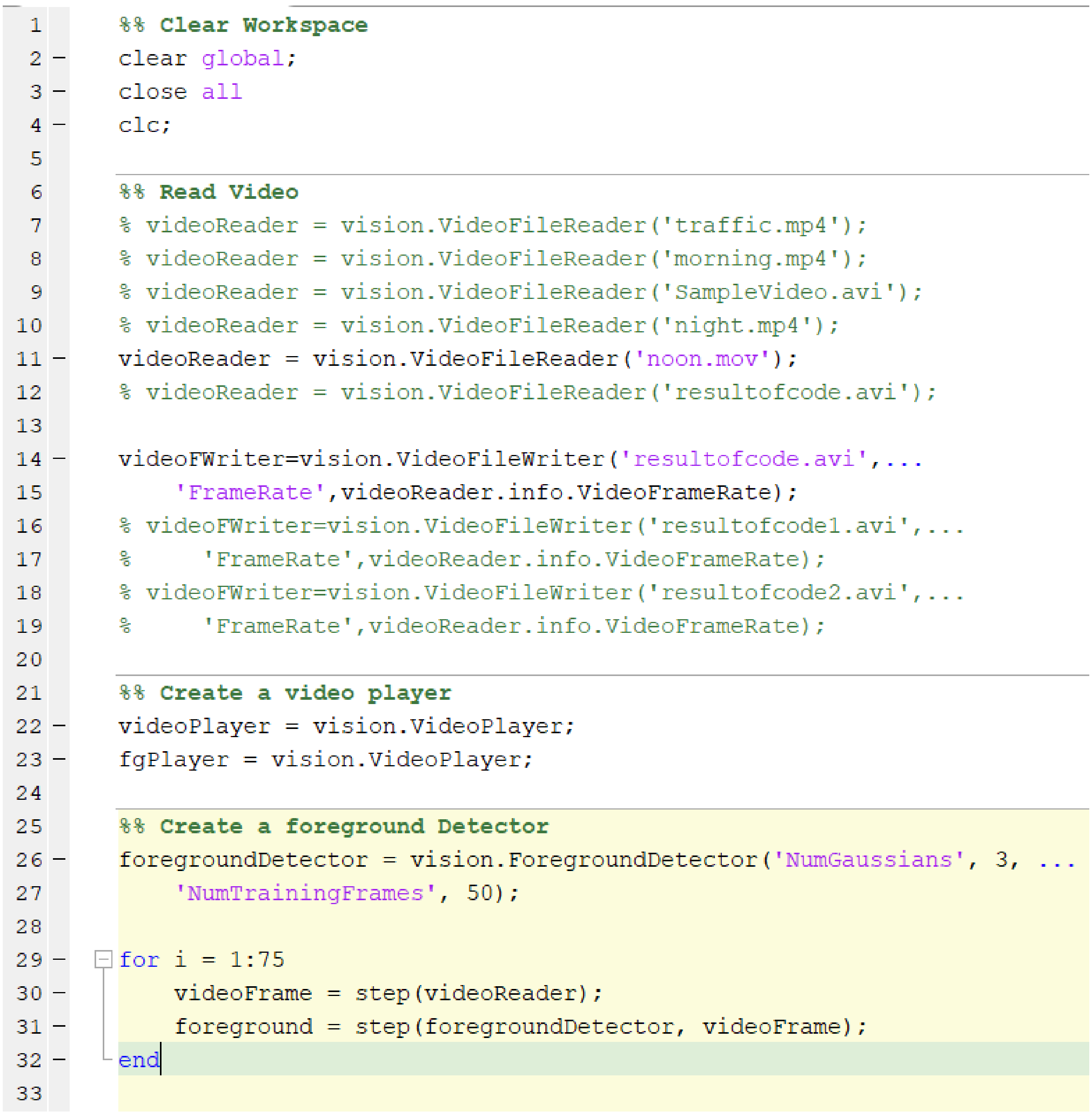
```
corr =

   -0.0184
```

Figure 84 The Correlation coefficient of the Embedded and the Extracted Watermark after the addition of the noise

Appendix B

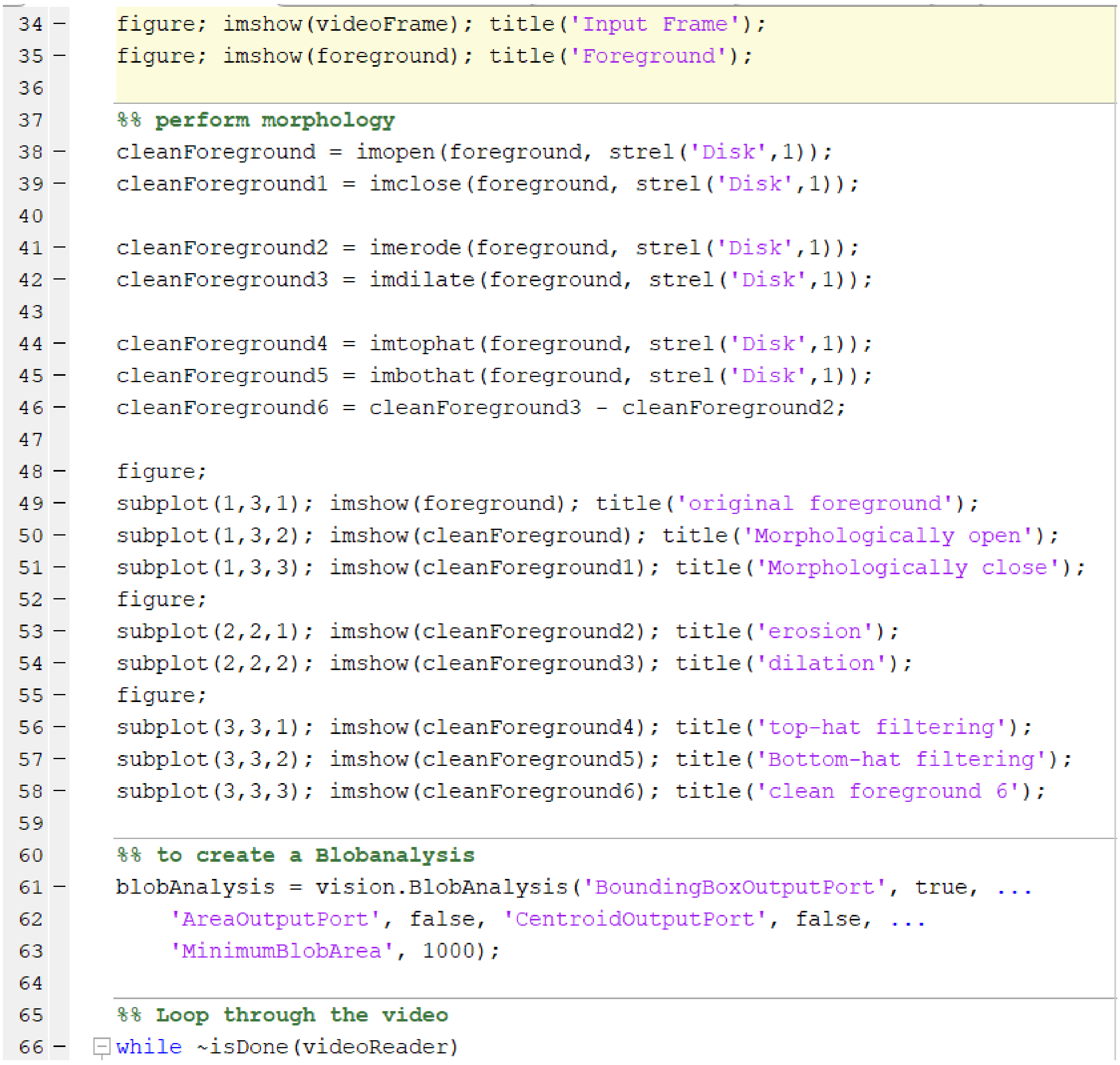
This section will include the code used to produce all the results so far,

1) **For the “Moving” Object Identification and the “Moving” Object Tracking:**



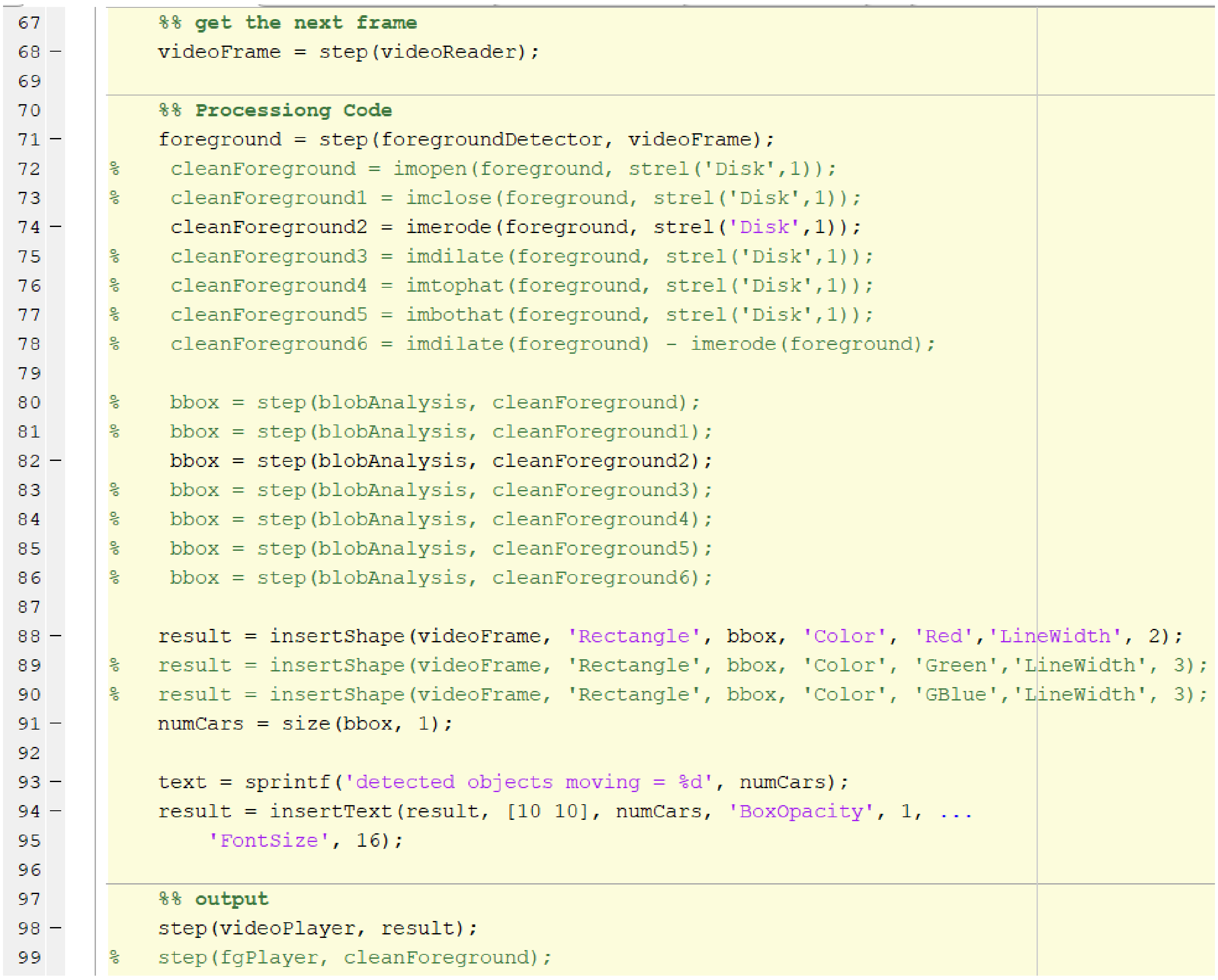
```matlab
%% Clear Workspace
clear global;
close all
clc;

%% Read Video
% videoReader = vision.VideoFileReader('traffic.mp4');
% videoReader = vision.VideoFileReader('morning.mp4');
% videoReader = vision.VideoFileReader('SampleVideo.avi');
% videoReader = vision.VideoFileReader('night.mp4');
videoReader = vision.VideoFileReader('noon.mov');
% videoReader = vision.VideoFileReader('resultofcode.avi');

videoFWriter=vision.VideoFileWriter('resultofcode.avi',...
    'FrameRate',videoReader.info.VideoFrameRate);
% videoFWriter=vision.VideoFileWriter('resultofcode1.avi',...
%     'FrameRate',videoReader.info.VideoFrameRate);
% videoFWriter=vision.VideoFileWriter('resultofcode2.avi',...
%     'FrameRate',videoReader.info.VideoFrameRate);

%% Create a video player
videoPlayer = vision.VideoPlayer;
fgPlayer = vision.VideoPlayer;

%% Create a foreground Detector
foregroundDetector = vision.ForegroundDetector('NumGaussians', 3, ...
    'NumTrainingFrames', 50);

for i = 1:75
    videoFrame = step(videoReader);
    foreground = step(foregroundDetector, videoFrame);
end

```

Figure 85 Moving object identification and the Moving Object Tracking Code (Part 1)



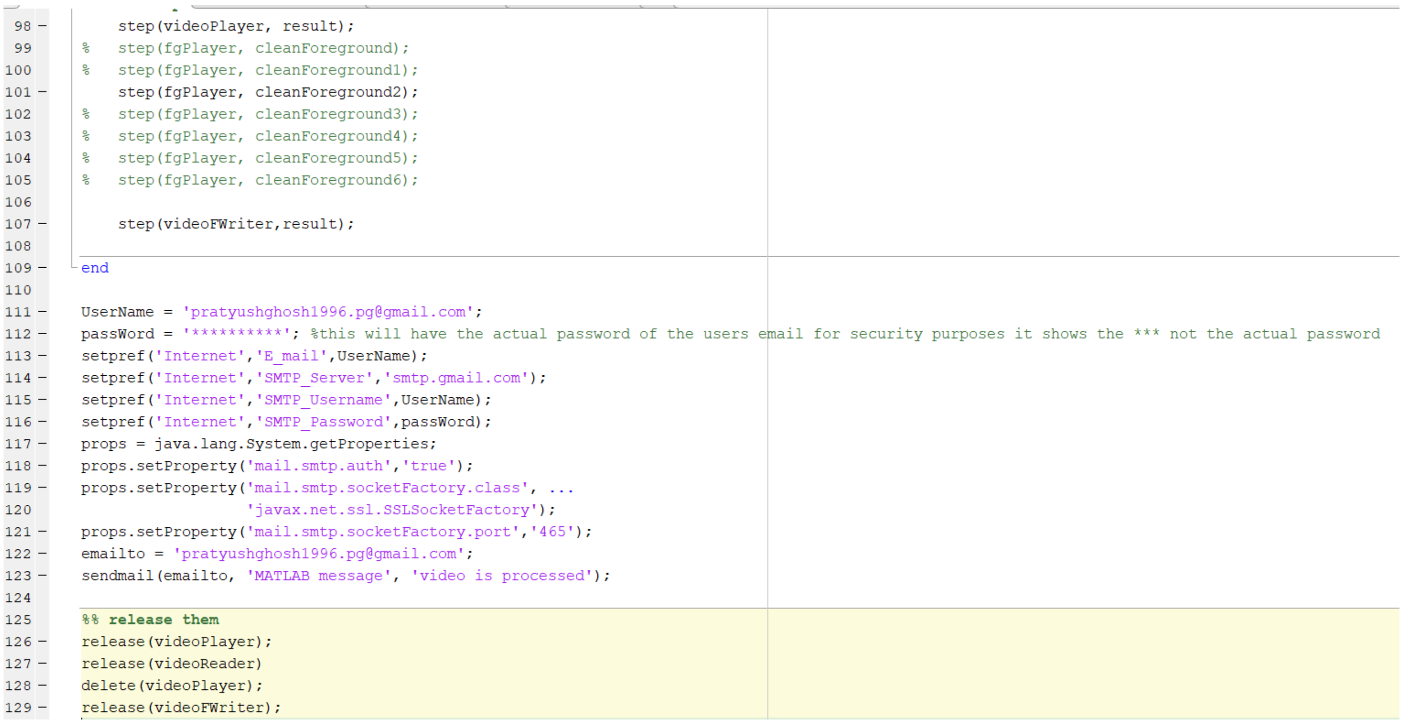
```
figure; imshow(videoFrame); title('Input Frame');
figure; imshow(foreground); title('Foreground');

%% perform morphology
cleanForeground = imopen(foreground, strel('Disk',1));
cleanForeground1 = imclose(foreground, strel('Disk',1));

cleanForeground2 = imerode(foreground, strel('Disk',1));
cleanForeground3 = imdilate(foreground, strel('Disk',1));

cleanForeground4 = imtophat(foreground, strel('Disk',1));
cleanForeground5 = imbothat(foreground, strel('Disk',1));
cleanForeground6 = cleanForeground3 - cleanForeground2;

figure;
subplot(1,3,1); imshow(foreground); title('original foreground');
subplot(1,3,2); imshow(cleanForeground); title('Morphologically open');
subplot(1,3,3); imshow(cleanForeground1); title('Morphologically close');
figure;
subplot(2,2,1); imshow(cleanForeground2); title('erosion');
subplot(2,2,2); imshow(cleanForeground3); title('dilation');
figure;
subplot(3,3,1); imshow(cleanForeground4); title('top-hat filtering');
subplot(3,3,2); imshow(cleanForeground5); title('Bottom-hat filtering');
subplot(3,3,3); imshow(cleanForeground6); title('clean foreground 6');

%% to create a Blobanalysis
blobAnalysis = vision.BlobAnalysis('BoundingBoxOutputPort', true, ...
    'AreaOutputPort', false, 'CentroidOutputPort', false, ...
    'MinimumBlobArea', 1000);

%% Loop through the video
while ~isDone(videoReader)
```

Figure 86 Moving Object Identification and the Moving Object Tracking Code (Part 2)



```matlab
    %% get the next frame
    videoFrame = step(videoReader);

    %% Processiong Code
    foreground = step(foregroundDetector, videoFrame);
%     cleanForeground = imopen(foreground, strel('Disk',1));
%     cleanForeground1 = imclose(foreground, strel('Disk',1));
     cleanForeground2 = imerode(foreground, strel('Disk',1));
%     cleanForeground3 = imdilate(foreground, strel('Disk',1));
%     cleanForeground4 = imtophat(foreground, strel('Disk',1));
%     cleanForeground5 = imbothat(foreground, strel('Disk',1));
%     cleanForeground6 = imdilate(foreground) - imerode(foreground);

%    bbox = step(blobAnalysis, cleanForeground);
%    bbox = step(blobAnalysis, cleanForeground1);
     bbox = step(blobAnalysis, cleanForeground2);
%    bbox = step(blobAnalysis, cleanForeground3);
%    bbox = step(blobAnalysis, cleanForeground4);
%    bbox = step(blobAnalysis, cleanForeground5);
%    bbox = step(blobAnalysis, cleanForeground6);

    result = insertShape(videoFrame, 'Rectangle', bbox, 'Color', 'Red','LineWidth', 2);
%   result = insertShape(videoFrame, 'Rectangle', bbox, 'Color', 'Green','LineWidth', 3);
%   result = insertShape(videoFrame, 'Rectangle', bbox, 'Color', 'GBlue','LineWidth', 3);
    numCars = size(bbox, 1);

    text = sprintf('detected objects moving = %d', numCars);
    result = insertText(result, [10 10], numCars, 'BoxOpacity', 1, ...
        'FontSize', 16);

    %% output
    step(videoPlayer, result);
%   step(fgPlayer, cleanForeground);
```

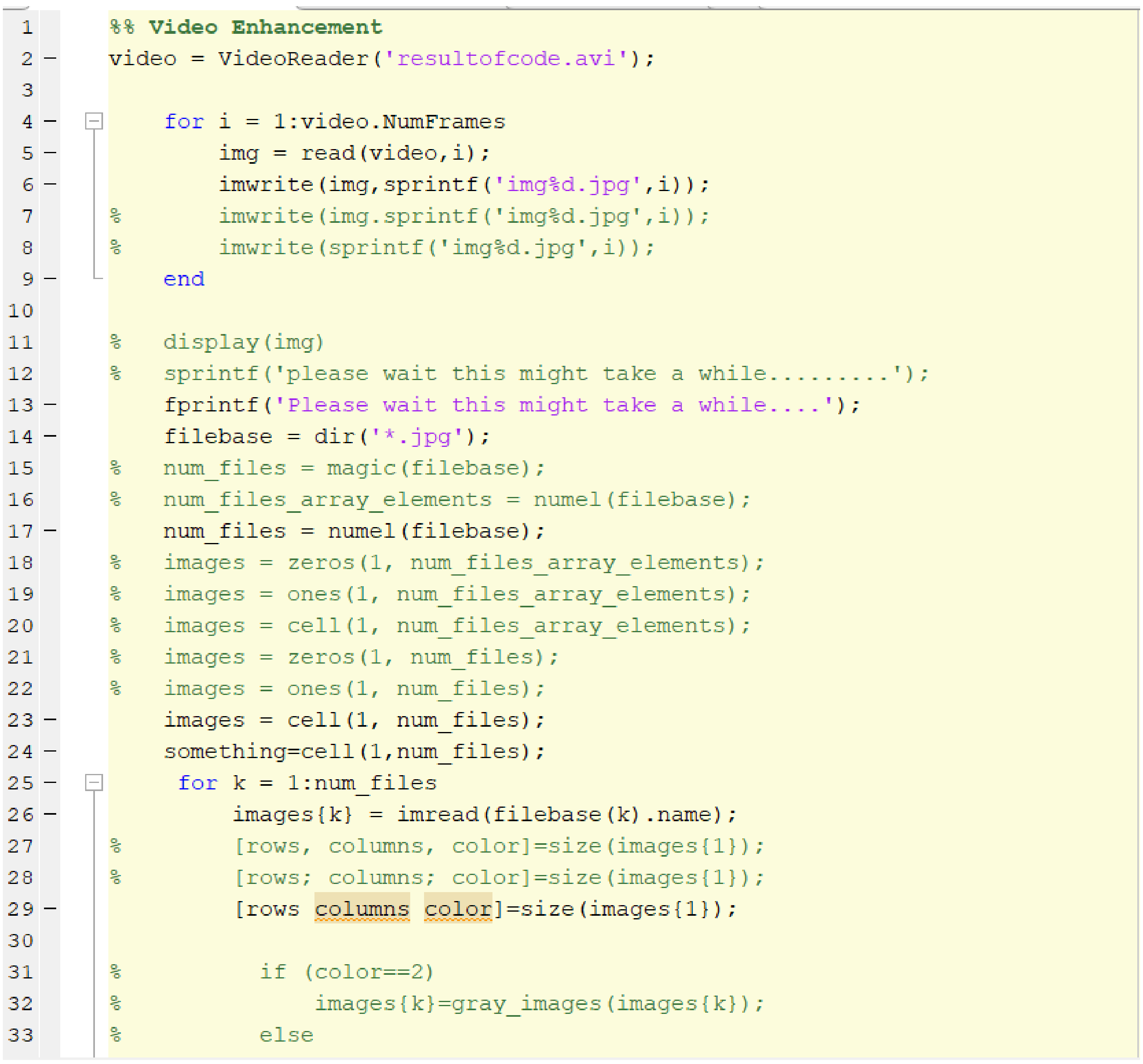
Figure 87 Moving Object Identification and the Moving Object Tracking Code (Part 3)



```matlab
    step(videoPlayer, result);
%   step(fgPlayer, cleanForeground);
%   step(fgPlayer, cleanForeground1);
    step(fgPlayer, cleanForeground2);
%   step(fgPlayer, cleanForeground3);
%   step(fgPlayer, cleanForeground4);
%   step(fgPlayer, cleanForeground5);
%   step(fgPlayer, cleanForeground6);

    step(videoFWriter,result);

end

UserName = 'pratyushghosh1996.pg@gmail.com';
passWord = '**********'; %this will have the actual password of the users email for security purposes it shows the *** not the actual password
setpref('Internet','E_mail',UserName);
setpref('Internet','SMTP_Server','smtp.gmail.com');
setpref('Internet','SMTP_Username',UserName);
setpref('Internet','SMTP_Password',passWord);
props = java.lang.System.getProperties;
props.setProperty('mail.smtp.auth','true');
props.setProperty('mail.smtp.socketFactory.class', ...
                  'javax.net.ssl.SSLSocketFactory');
props.setProperty('mail.smtp.socketFactory.port','465');
emailto = 'pratyushghosh1996.pg@gmail.com';
sendmail(emailto, 'MATLAB message', 'video is processed');

%% release them
release(videoPlayer);
release(videoReader)
delete(videoPlayer);
release(videoFWriter);
```

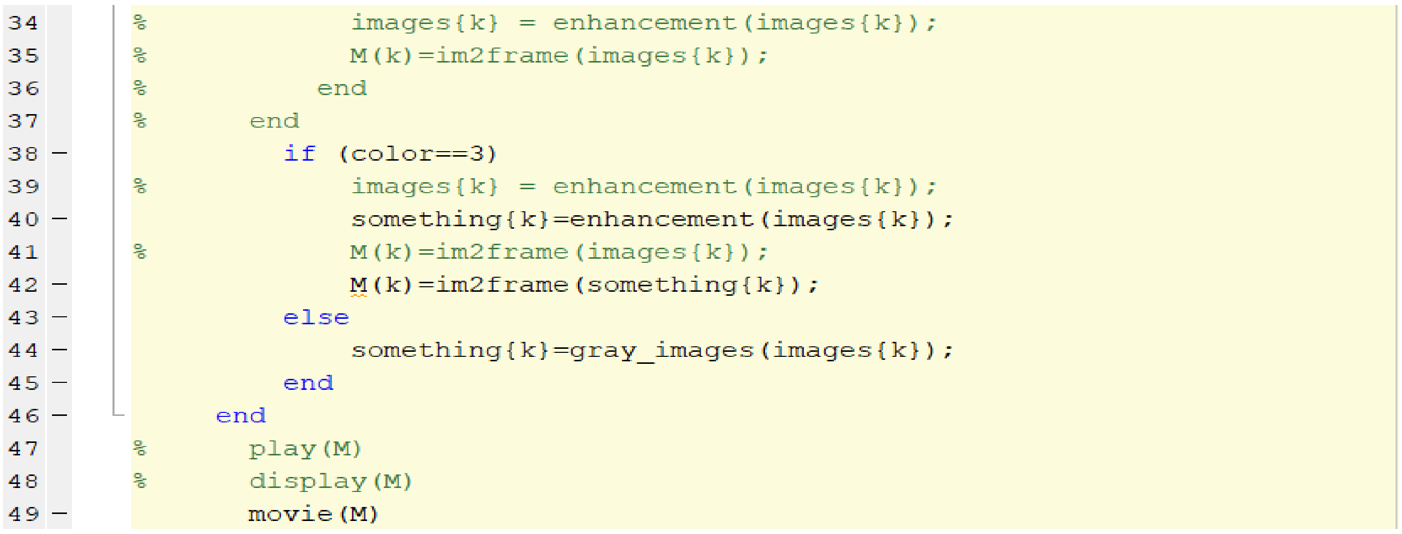
Figure 88 Moving Object Identification and the Moving Object Tracking Code (Part 4)

2) For the video quality enhancement



```
%% Video Enhancement
video = VideoReader('resultofcode.avi');

    for i = 1:video.NumFrames
        img = read(video,i);
        imwrite(img,sprintf('img%d.jpg',i));
%       imwrite(img.sprintf('img%d.jpg',i));
%       imwrite(sprintf('img%d.jpg',i));
    end

%   display(img)
%   sprintf('please wait this might take a while.........');
    fprintf('Please wait this might take a while....');
    filebase = dir('*.jpg');
%   num_files = magic(filebase);
%   num_files_array_elements = numel(filebase);
    num_files = numel(filebase);
%   images = zeros(1, num_files_array_elements);
%   images = ones(1, num_files_array_elements);
%   images = cell(1, num_files_array_elements);
%   images = zeros(1, num_files);
%   images = ones(1, num_files);
    images = cell(1, num_files);
    something=cell(1,num_files);
     for k = 1:num_files
         images{k} = imread(filebase(k).name);
%        [rows, columns, color]=size(images{1});
%        [rows; columns; color]=size(images{1});
         [rows columns color]=size(images{1});

%          if (color==2)
%              images{k}=gray_images(images{k});
%          else
```

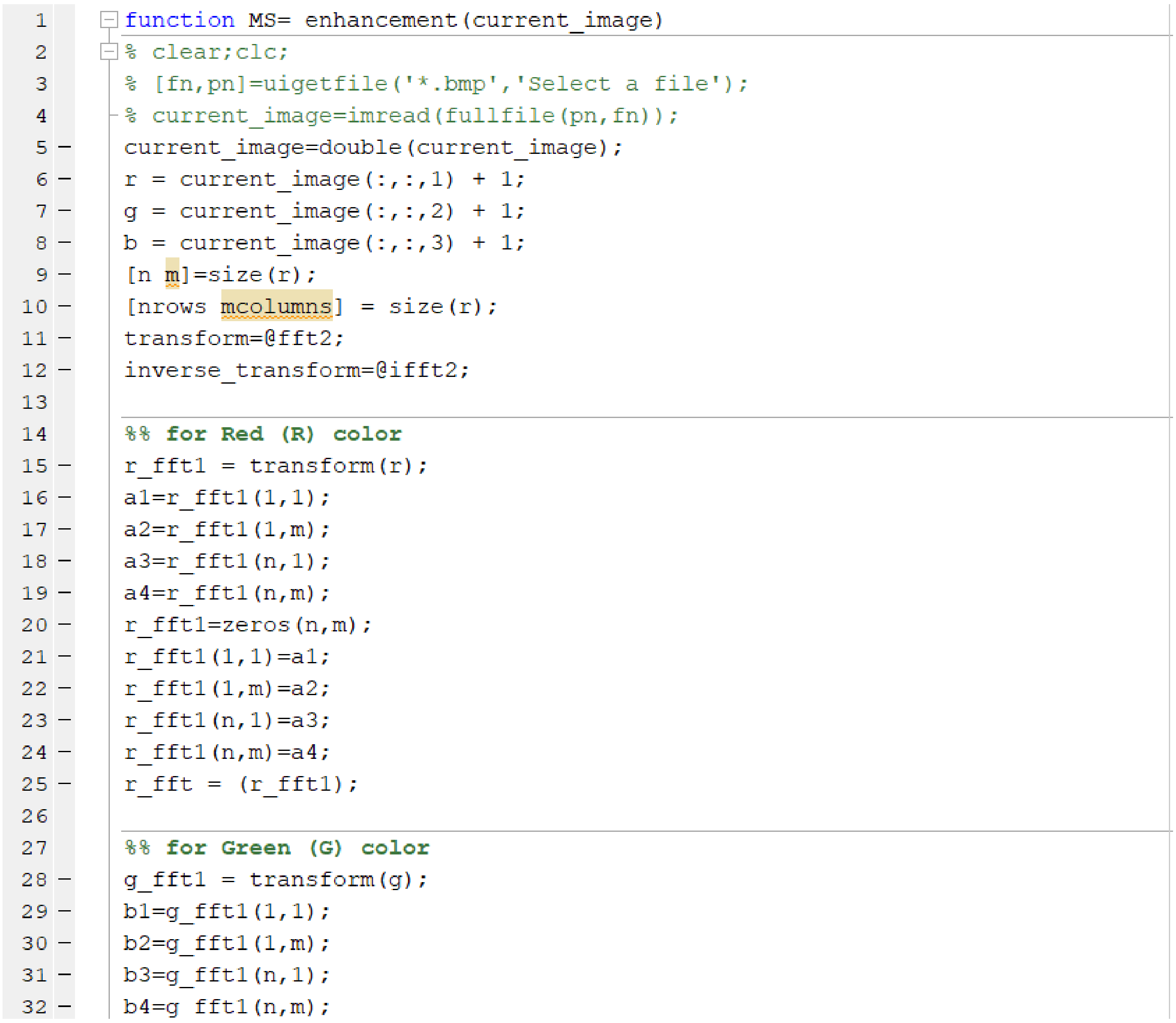
Figure 89 Video Quality Enhancement Code (Part 1)



```
%               images{k} = enhancement(images{k});
%               M(k)=im2frame(images{k});
%             end
%         end
            if (color==3)
%               images{k} = enhancement(images{k});
                something{k}=enhancement(images{k});
%               M(k)=im2frame(images{k});
                M(k)=im2frame(something{k});
            else
                something{k}=gray_images(images{k});
            end
        end
%         play(M)
%         display(M)
          movie(M)
```

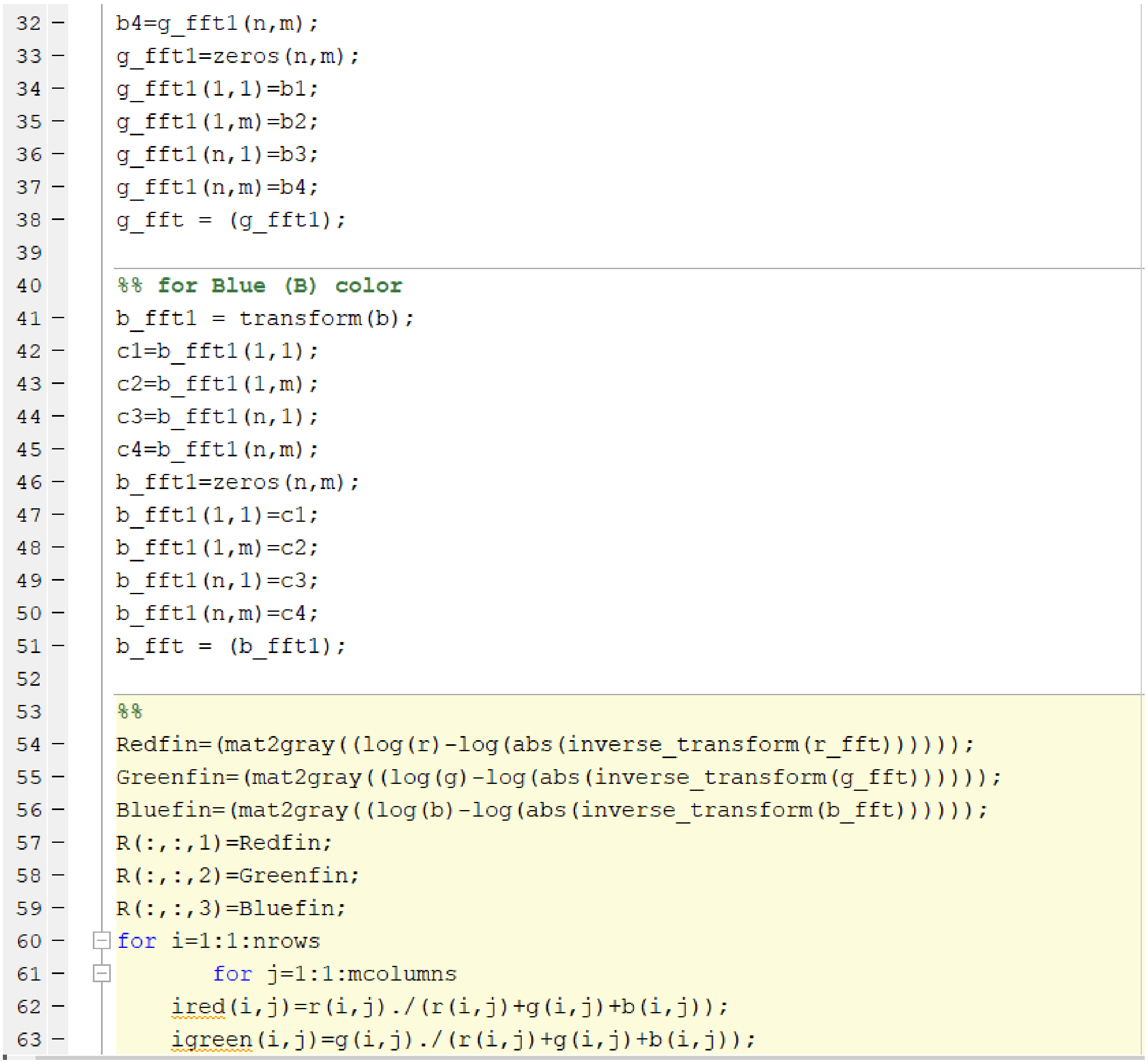
Figure 90 Video Quality Enhancement Code (Part 2)

a) The Enhancement Function (used for Color Formats)



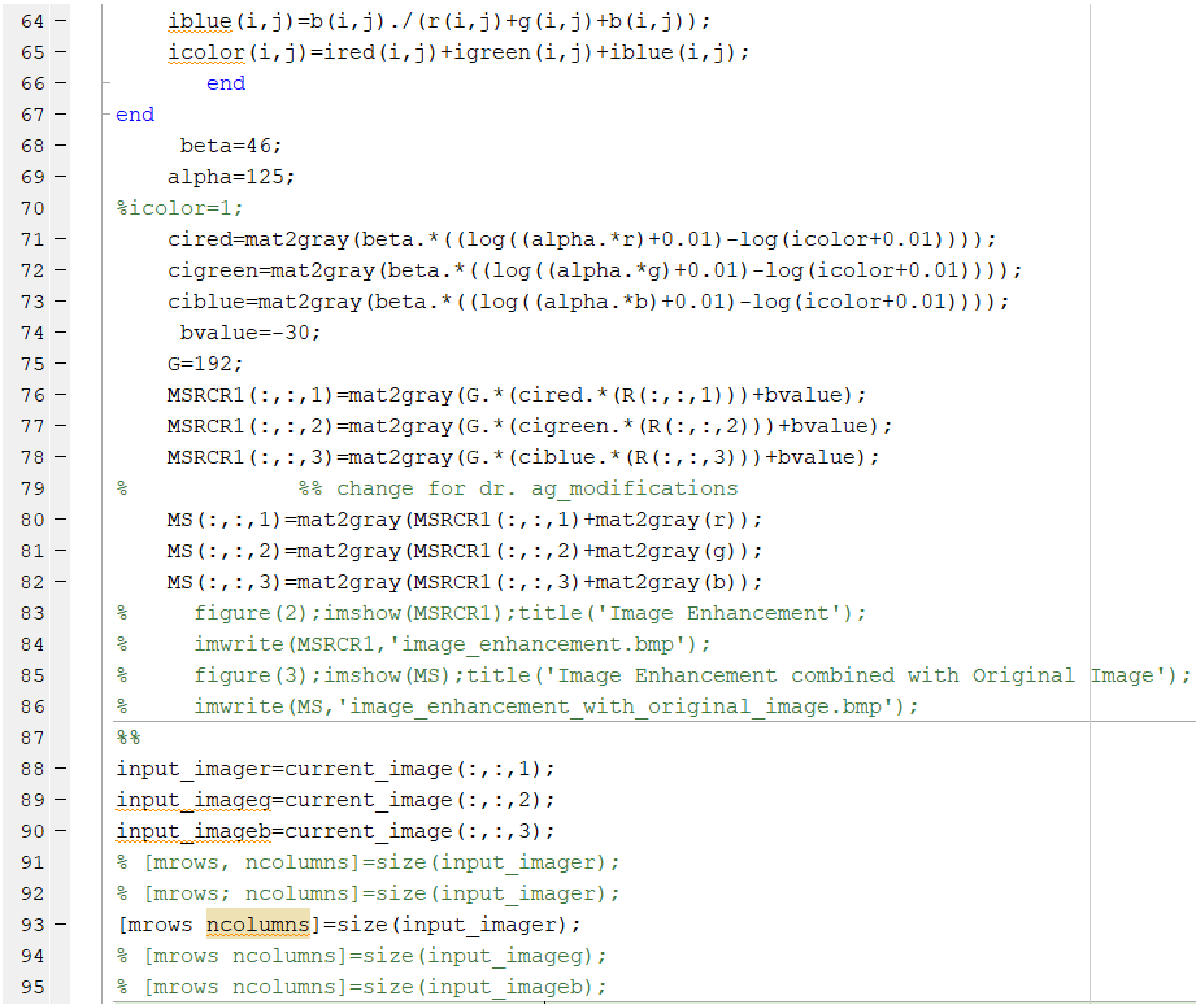
```
function MS= enhancement(current_image)
% clear;clc;
% [fn,pn]=uigetfile('*.bmp','Select a file');
% current_image=imread(fullfile(pn,fn));
current_image=double(current_image);
r = current_image(:,:,1) + 1;
g = current_image(:,:,2) + 1;
b = current_image(:,:,3) + 1;
[n m]=size(r);
[nrows mcolumns] = size(r);
transform=@fft2;
inverse_transform=@ifft2;

%% for Red (R) color
r_fft1 = transform(r);
a1=r_fft1(1,1);
a2=r_fft1(1,m);
a3=r_fft1(n,1);
a4=r_fft1(n,m);
r_fft1=zeros(n,m);
r_fft1(1,1)=a1;
r_fft1(1,m)=a2;
r_fft1(n,1)=a3;
r_fft1(n,m)=a4;
r_fft = (r_fft1);

%% for Green (G) color
g_fft1 = transform(g);
b1=g_fft1(1,1);
b2=g_fft1(1,m);
b3=g_fft1(n,1);
b4=g_fft1(n,m);
```

Figure 91 Enhancement Function Code (used for Color Formats) [Part 1]



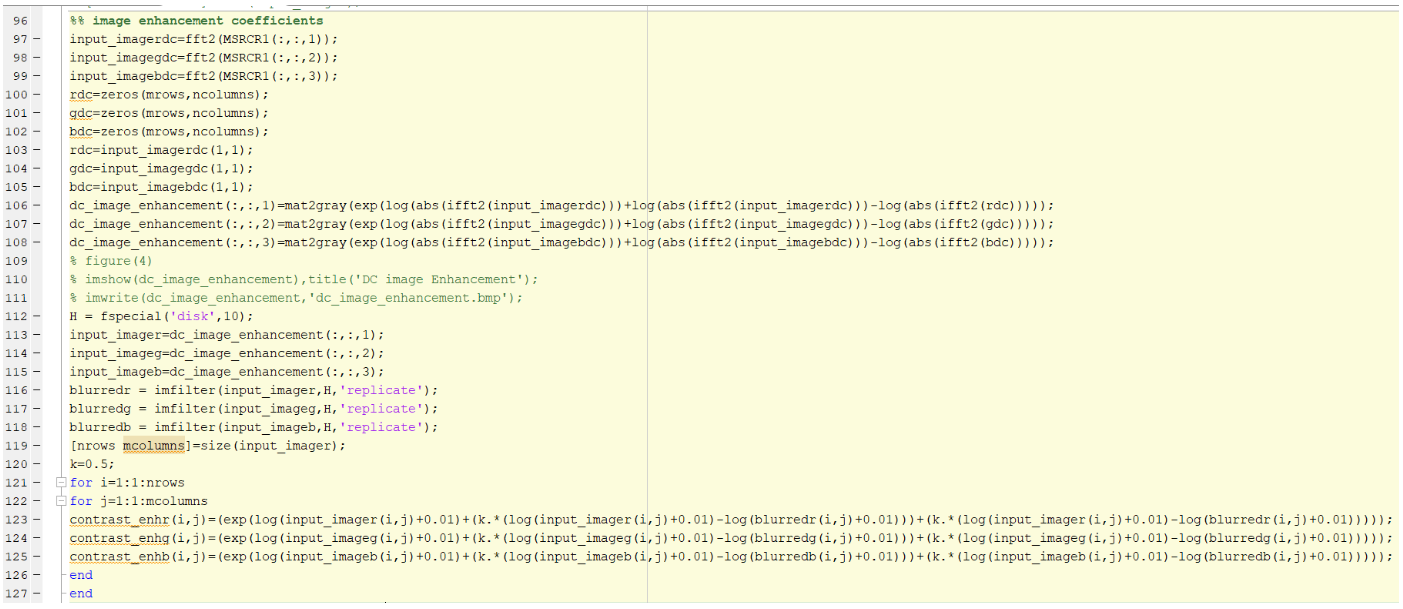
```
b4=g_fft1(n,m);
g_fft1=zeros(n,m);
g_fft1(1,1)=b1;
g_fft1(1,m)=b2;
g_fft1(n,1)=b3;
g_fft1(n,m)=b4;
g_fft = (g_fft1);

%% for Blue (B) color
b_fft1 = transform(b);
c1=b_fft1(1,1);
c2=b_fft1(1,m);
c3=b_fft1(n,1);
c4=b_fft1(n,m);
b_fft1=zeros(n,m);
b_fft1(1,1)=c1;
b_fft1(1,m)=c2;
b_fft1(n,1)=c3;
b_fft1(n,m)=c4;
b_fft = (b_fft1);

%%
Redfin=(mat2gray((log(r)-log(abs(inverse_transform(r_fft))))));
Greenfin=(mat2gray((log(g)-log(abs(inverse_transform(g_fft))))));
Bluefin=(mat2gray((log(b)-log(abs(inverse_transform(b_fft))))));
R(:,:,1)=Redfin;
R(:,:,2)=Greenfin;
R(:,:,3)=Bluefin;
for i=1:1:nrows
       for j=1:1:mcolumns
    ired(i,j)=r(i,j)./(r(i,j)+g(i,j)+b(i,j));
    igreen(i,j)=g(i,j)./(r(i,j)+g(i,j)+b(i,j));
```

Figure 92 Enhancement Function Code (used for Color Formats) [Part 2]



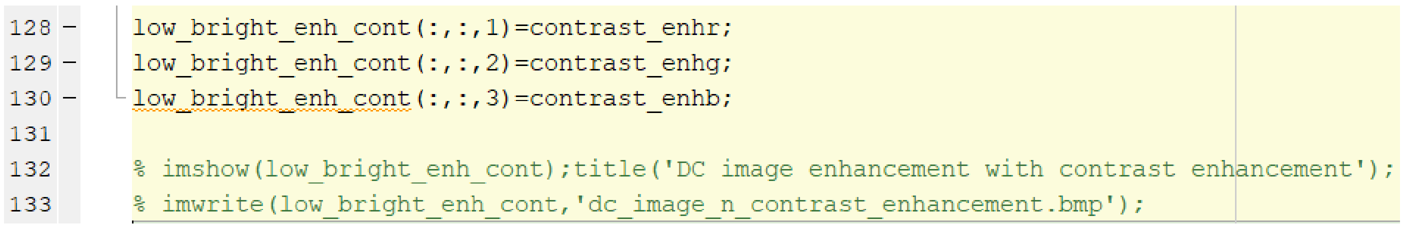
```
    iblue(i,j)=b(i,j)./(r(i,j)+g(i,j)+b(i,j));
    icolor(i,j)=ired(i,j)+igreen(i,j)+iblue(i,j);
      end
end
     beta=46;
    alpha=125;
%icolor=1;
    cired=mat2gray(beta.*((log((alpha.*r)+0.01)-log(icolor+0.01))));
    cigreen=mat2gray(beta.*((log((alpha.*g)+0.01)-log(icolor+0.01))));
    ciblue=mat2gray(beta.*((log((alpha.*b)+0.01)-log(icolor+0.01))));
     bvalue=-30;
    G=192;
    MSRCR1(:,:,1)=mat2gray(G.*(cired.*(R(:,:,1)))+bvalue);
    MSRCR1(:,:,2)=mat2gray(G.*(cigreen.*(R(:,:,2)))+bvalue);
    MSRCR1(:,:,3)=mat2gray(G.*(ciblue.*(R(:,:,3)))+bvalue);
%              %% change for dr. ag_modifications
    MS(:,:,1)=mat2gray(MSRCR1(:,:,1)+mat2gray(r));
    MS(:,:,2)=mat2gray(MSRCR1(:,:,2)+mat2gray(g));
    MS(:,:,3)=mat2gray(MSRCR1(:,:,3)+mat2gray(b));
%     figure(2);imshow(MSRCR1);title('Image Enhancement');
%     imwrite(MSRCR1,'image_enhancement.bmp');
%     figure(3);imshow(MS);title('Image Enhancement combined with Original Image');
%     imwrite(MS,'image_enhancement_with_original_image.bmp');
%%
input_imager=current_image(:,:,1);
input_imageg=current_image(:,:,2);
input_imageb=current_image(:,:,3);
% [mrows, ncolumns]=size(input_imager);
% [mrows; ncolumns]=size(input_imager);
[mrows ncolumns]=size(input_imager);
% [mrows ncolumns]=size(input_imageg);
% [mrows ncolumns]=size(input_imageb);
```

Figure 93 Enhancement Function Code (used for Color Formats) [Part 3]



```
%% image enhancement coefficients
input_imagerdc=fft2(MSRCR1(:,:,1));
input_imagegdc=fft2(MSRCR1(:,:,2));
input_imagebdc=fft2(MSRCR1(:,:,3));
rdc=zeros(mrows,ncolumns);
gdc=zeros(mrows,ncolumns);
bdc=zeros(mrows,ncolumns);
rdc=input_imagerdc(1,1);
gdc=input_imagegdc(1,1);
bdc=input_imagebdc(1,1);
dc_image_enhancement(:,:,1)=mat2gray(exp(log(abs(ifft2(input_imagerdc)))+log(abs(ifft2(input_imagerdc)))-log(abs(ifft2(rdc)))));
dc_image_enhancement(:,:,2)=mat2gray(exp(log(abs(ifft2(input_imagegdc)))+log(abs(ifft2(input_imagegdc)))-log(abs(ifft2(gdc)))));
dc_image_enhancement(:,:,3)=mat2gray(exp(log(abs(ifft2(input_imagebdc)))+log(abs(ifft2(input_imagebdc)))-log(abs(ifft2(bdc)))));
% figure(4)
% imshow(dc_image_enhancement),title('DC image Enhancement');
% imwrite(dc_image_enhancement,'dc_image_enhancement.bmp');
H = fspecial('disk',10);
input_imager=dc_image_enhancement(:,:,1);
input_imageg=dc_image_enhancement(:,:,2);
input_imageb=dc_image_enhancement(:,:,3);
blurredr = imfilter(input_imager,H,'replicate');
blurredg = imfilter(input_imageg,H,'replicate');
blurredb = imfilter(input_imageb,H,'replicate');
[nrows mcolumns]=size(input_imager);
k=0.5;
for i=1:1:nrows
for j=1:1:mcolumns
contrast_enhr(i,j)=(exp(log(input_imager(i,j)+0.01)+(k.*(log(input_imager(i,j)+0.01)-log(blurredr(i,j)+0.01)))+(k.*(log(input_imager(i,j)+0.01)-log(blurredr(i,j)+0.01)))));
contrast_enhg(i,j)=(exp(log(input_imageg(i,j)+0.01)+(k.*(log(input_imageg(i,j)+0.01)-log(blurredg(i,j)+0.01)))+(k.*(log(input_imageg(i,j)+0.01)-log(blurredg(i,j)+0.01)))));
contrast_enhb(i,j)=(exp(log(input_imageb(i,j)+0.01)+(k.*(log(input_imageb(i,j)+0.01)-log(blurredb(i,j)+0.01)))+(k.*(log(input_imageb(i,j)+0.01)-log(blurredb(i,j)+0.01)))));
end
end
```

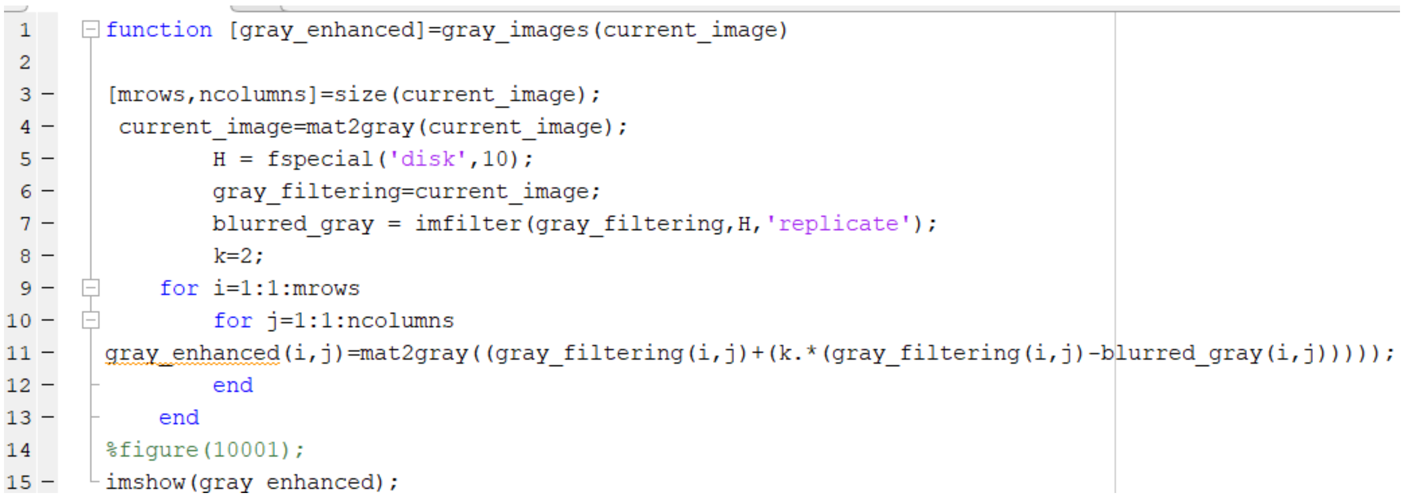
Figure 94 Enhancement Function Code (used for Color Formats) [Part 4]



```
low_bright_enh_cont(:,:,1)=contrast_enhr;
low_bright_enh_cont(:,:,2)=contrast_enhg;
low_bright_enh_cont(:,:,3)=contrast_enhb;

% imshow(low_bright_enh_cont);title('DC image enhancement with contrast enhancement');
% imwrite(low_bright_enh_cont,'dc_image_n_contrast_enhancement.bmp');
```

Figure 95 Enhancement Function Code (used for Color Formats) [Part 5]

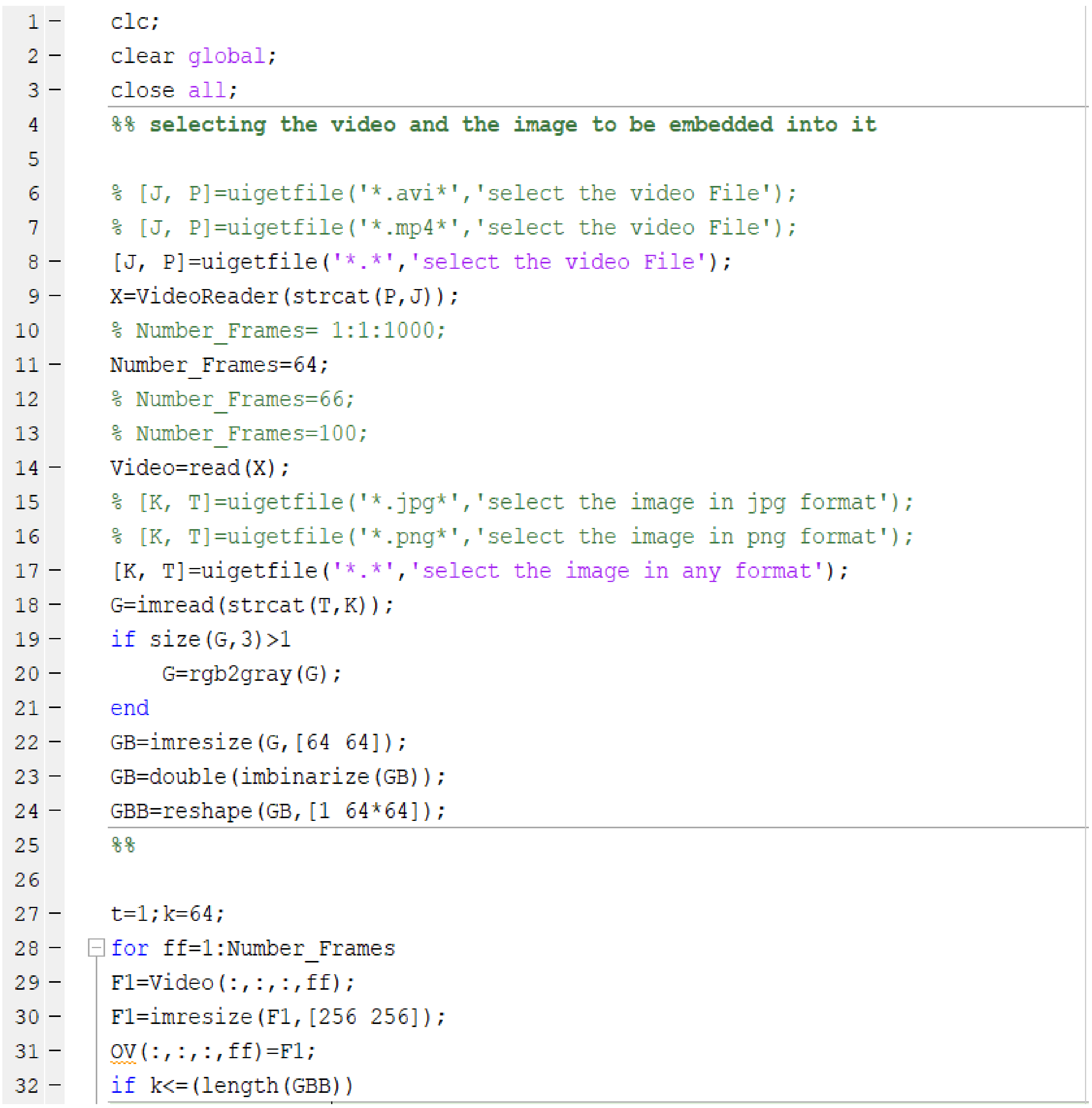
b) The Enhancement Function (used for Grayscale and Black and White Formats)



```
function [gray_enhanced]=gray_images(current_image)

[mrows,ncolumns]=size(current_image);
 current_image=mat2gray(current_image);
        H = fspecial('disk',10);
        gray_filtering=current_image;
        blurred_gray = imfilter(gray_filtering,H,'replicate');
        k=2;
    for i=1:1:mrows
        for j=1:1:ncolumns
gray_enhanced(i,j)=mat2gray((gray_filtering(i,j)+(k.*(gray_filtering(i,j)-blurred_gray(i,j)))));
        end
    end
%figure(10001);
imshow(gray_enhanced);
```

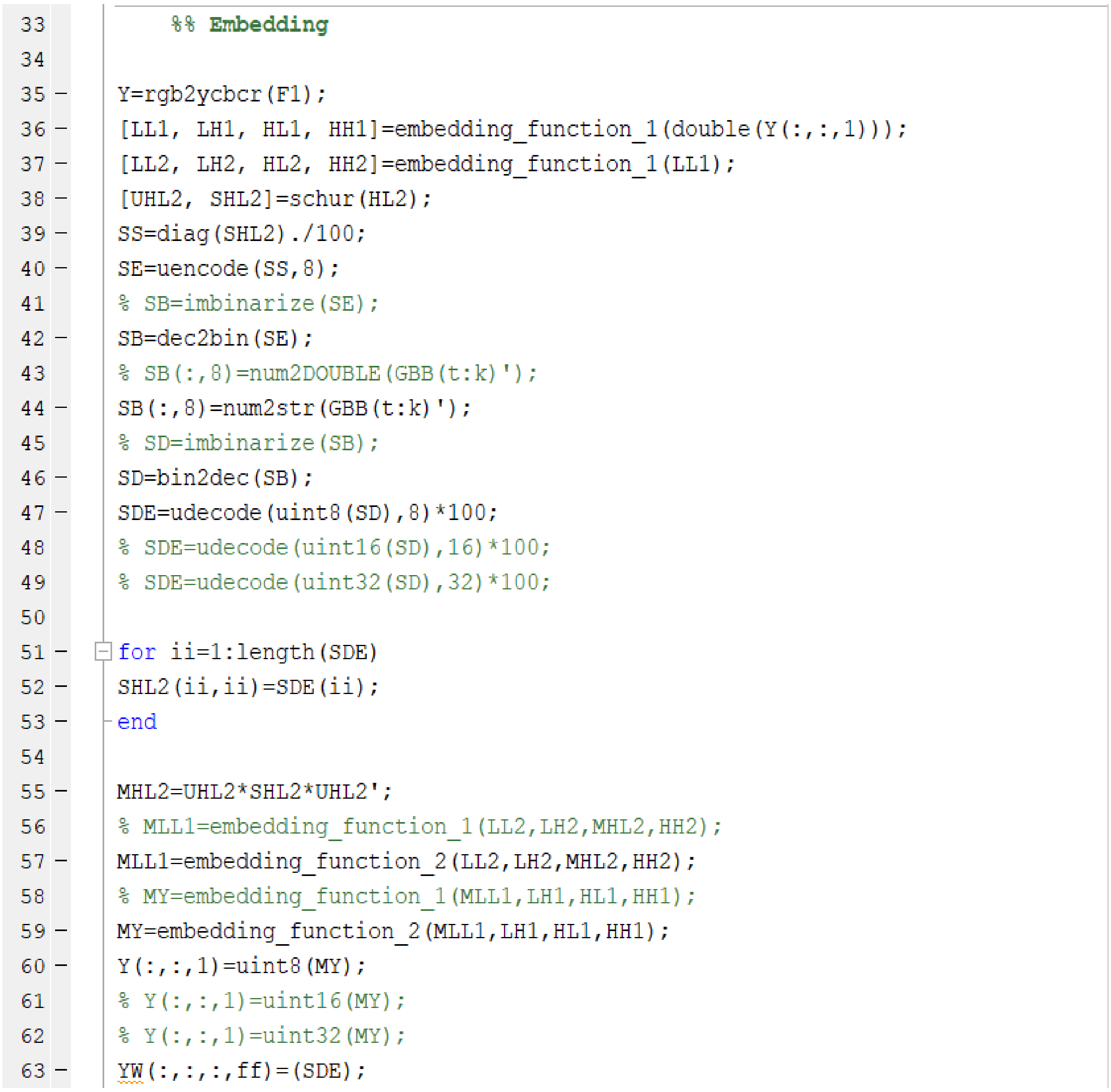
Figure 96 Enhancement Function Code (used for Grayscale and Black and White Formats)

3) For the Discrete Wavelet Transform based Watermarking



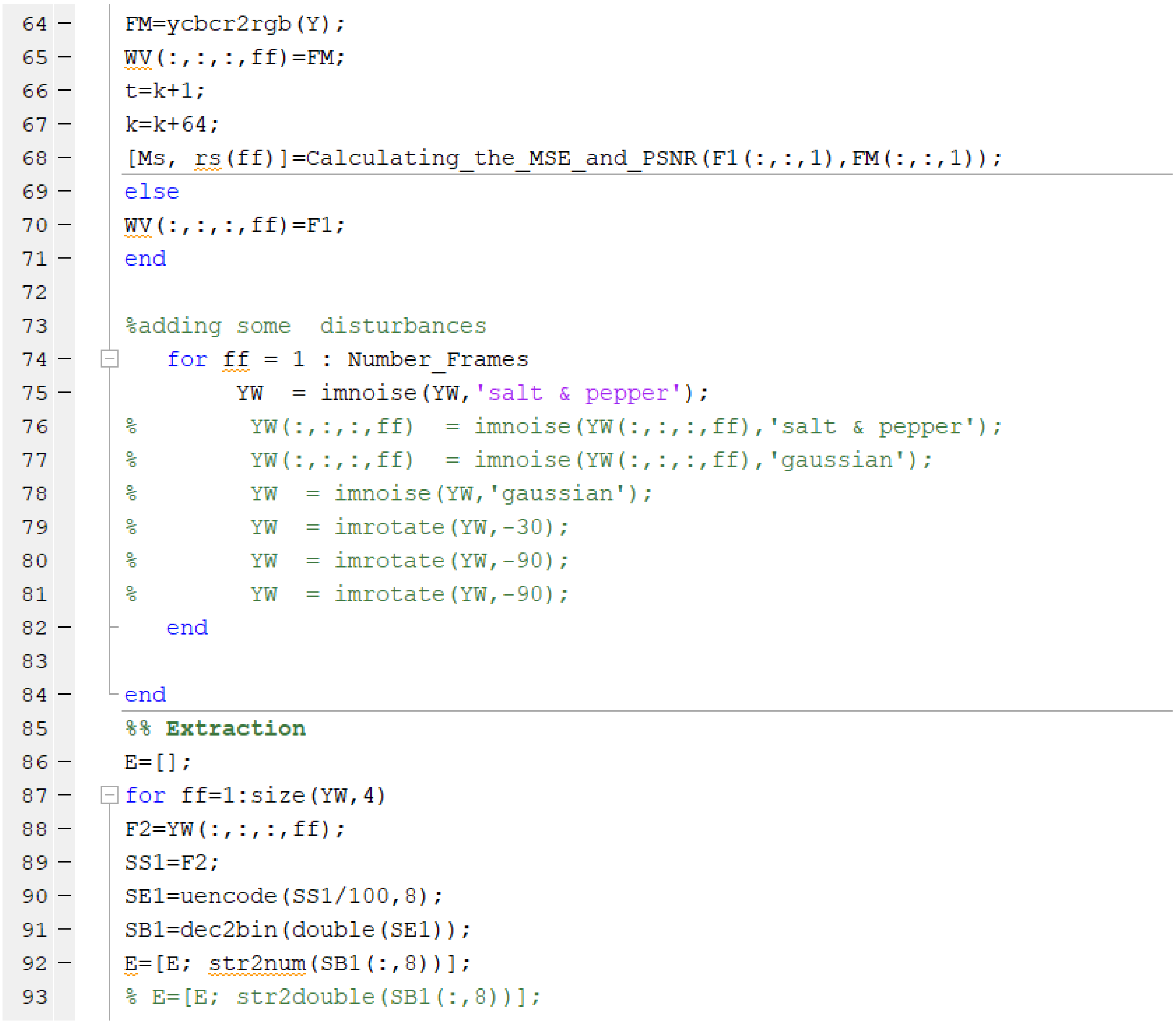
```
clc;
clear global;
close all;
%% selecting the video and the image to be embedded into it

% [J, P]=uigetfile('*.avi*','select the video File');
% [J, P]=uigetfile('*.mp4*','select the video File');
[J, P]=uigetfile('*.*','select the video File');
X=VideoReader(strcat(P,J));
% Number_Frames= 1:1:1000;
Number_Frames=64;
% Number_Frames=66;
% Number_Frames=100;
Video=read(X);
% [K, T]=uigetfile('*.jpg*','select the image in jpg format');
% [K, T]=uigetfile('*.png*','select the image in png format');
[K, T]=uigetfile('*.*','select the image in any format');
G=imread(strcat(T,K));
if size(G,3)>1
    G=rgb2gray(G);
end
GB=imresize(G,[64 64]);
GB=double(imbinarize(GB));
GBB=reshape(GB,[1 64*64]);
%%

t=1;k=64;
for ff=1:Number_Frames
F1=Video(:,:,:,ff);
F1=imresize(F1,[256 256]);
OV(:,:,:,ff)=F1;
if k<=(length(GBB))
```

Figure 97 The Discrete Wavelet Transform based Watermarking Code (Part 1)



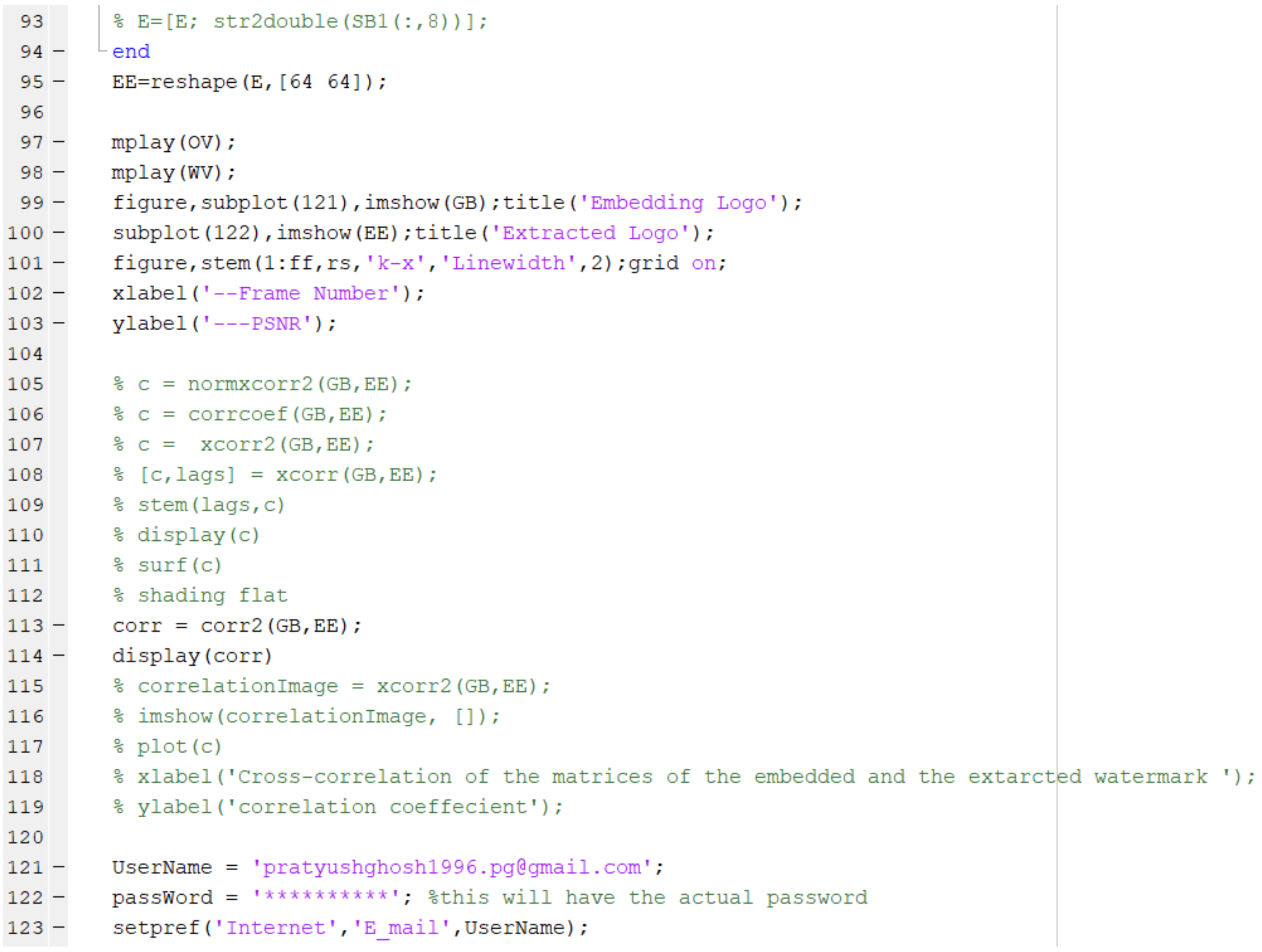
```
    %% Embedding

Y=rgb2ycbcr(F1);
[LL1, LH1, HL1, HH1]=embedding_function_1(double(Y(:,:,1)));
[LL2, LH2, HL2, HH2]=embedding_function_1(LL1);
[UHL2, SHL2]=schur(HL2);
SS=diag(SHL2)./100;
SE=uencode(SS,8);
% SB=imbinarize(SE);
SB=dec2bin(SE);
% SB(:,8)=num2DOUBLE(GBB(t:k)');
SB(:,8)=num2str(GBB(t:k)');
% SD=imbinarize(SB);
SD=bin2dec(SB);
SDE=udecode(uint8(SD),8)*100;
% SDE=udecode(uint16(SD),16)*100;
% SDE=udecode(uint32(SD),32)*100;

for ii=1:length(SDE)
SHL2(ii,ii)=SDE(ii);
end

MHL2=UHL2*SHL2*UHL2';
% MLL1=embedding_function_1(LL2,LH2,MHL2,HH2);
MLL1=embedding_function_2(LL2,LH2,MHL2,HH2);
% MY=embedding_function_1(MLL1,LH1,HL1,HH1);
MY=embedding_function_2(MLL1,LH1,HL1,HH1);
Y(:,:,1)=uint8(MY);
% Y(:,:,1)=uint16(MY);
% Y(:,:,1)=uint32(MY);
YW(:,:,:,ff)=(SDE);
```

Figure 98 The Discrete Wavelet Transform based Watermarking Code (Part 2)



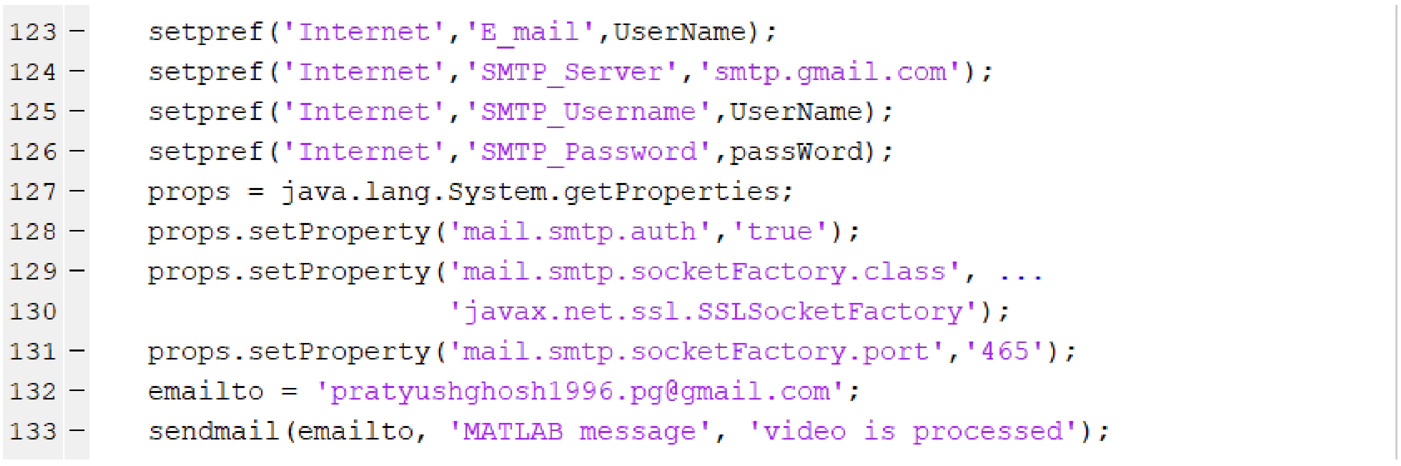
```
FM=ycbcr2rgb(Y);
WV(:,:,:,ff)=FM;
t=k+1;
k=k+64;
[Ms, rs(ff)]=Calculating_the_MSE_and_PSNR(F1(:,:,1),FM(:,:,1));
else
WV(:,:,:,ff)=F1;
end

%adding some  disturbances
   for ff = 1 : Number_Frames
        YW  = imnoise(YW,'salt & pepper');
%        YW(:,:,:,ff)  = imnoise(YW(:,:,:,ff),'salt & pepper');
%        YW(:,:,:,ff)  = imnoise(YW(:,:,:,ff),'gaussian');
%        YW  = imnoise(YW,'gaussian');
%        YW  = imrotate(YW,-30);
%        YW  = imrotate(YW,-90);
%        YW  = imrotate(YW,-90);
   end

end
%% Extraction
E=[];
for ff=1:size(YW,4)
F2=YW(:,:,:,ff);
SS1=F2;
SE1=uencode(SS1/100,8);
SB1=dec2bin(double(SE1));
E=[E; str2num(SB1(:,8))];
% E=[E; str2double(SB1(:,8))];
```

Figure 99 The Discrete Wavelet Transform based Watermarking Code (Part 3)



```matlab
  % E=[E; str2double(SB1(:,8))];
end
EE=reshape(E,[64 64]);

mplay(OV);
mplay(WV);
figure,subplot(121),imshow(GB);title('Embedding Logo');
subplot(122),imshow(EE);title('Extracted Logo');
figure,stem(1:ff,rs,'k-x','Linewidth',2);grid on;
xlabel('--Frame Number');
ylabel('---PSNR');

% c = normxcorr2(GB,EE);
% c = corrcoef(GB,EE);
% c =  xcorr2(GB,EE);
% [c,lags] = xcorr(GB,EE);
% stem(lags,c)
% display(c)
% surf(c)
% shading flat
corr = corr2(GB,EE);
display(corr)
% correlationImage = xcorr2(GB,EE);
% imshow(correlationImage, []);
% plot(c)
% xlabel('Cross-correlation of the matrices of the embedded and the extarcted watermark ');
% ylabel('correlation coeffecient');

UserName = 'pratyushghosh1996.pg@gmail.com';
passWord = '**********'; %this will have the actual password
setpref('Internet','E_mail',UserName);
```

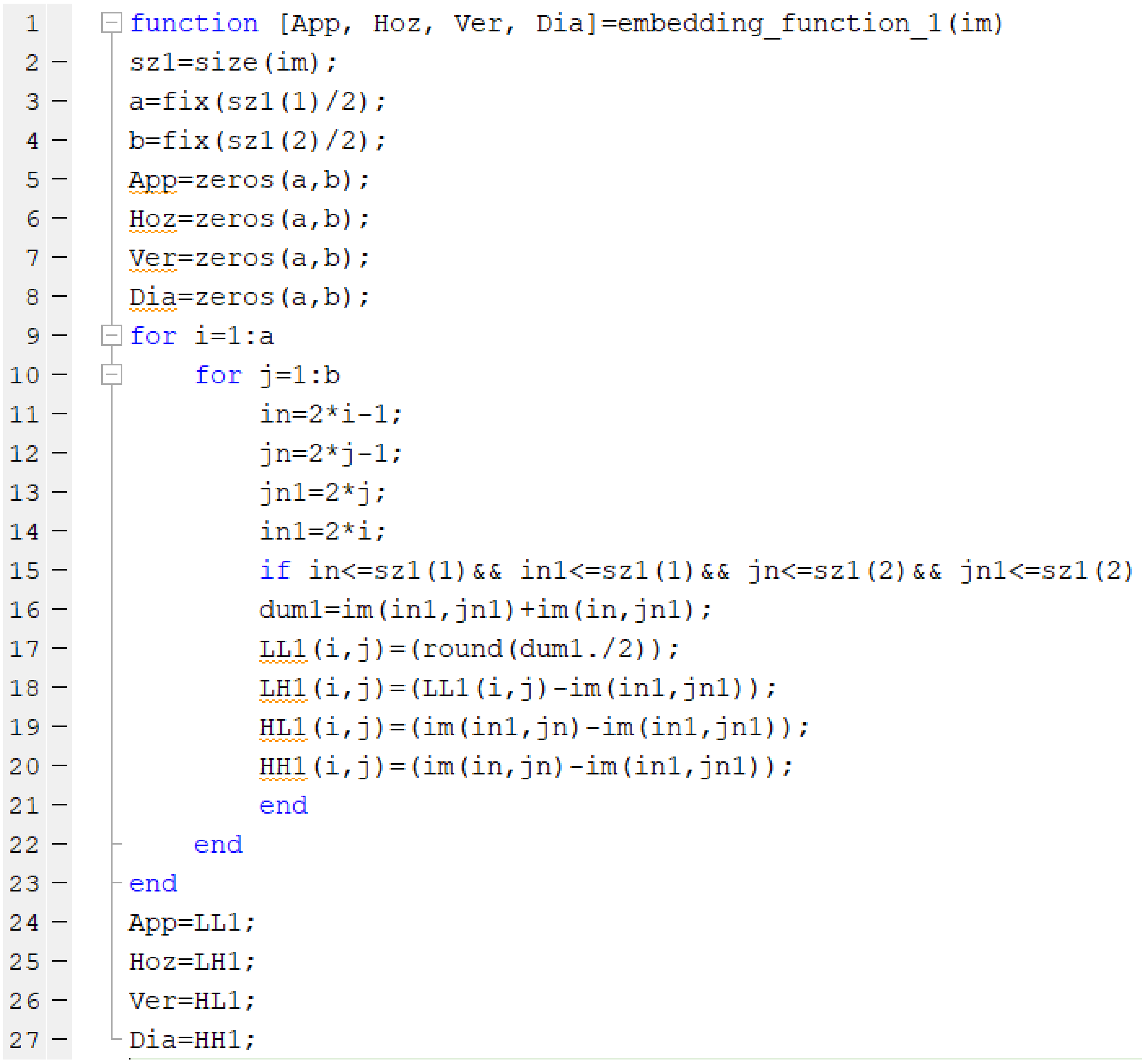
Figure 100 The Discrete Wavelet Transform based Watermarking Code (Part 4)



```matlab
setpref('Internet','E_mail',UserName);
setpref('Internet','SMTP_Server','smtp.gmail.com');
setpref('Internet','SMTP_Username',UserName);
setpref('Internet','SMTP_Password',passWord);
props = java.lang.System.getProperties;
props.setProperty('mail.smtp.auth','true');
props.setProperty('mail.smtp.socketFactory.class', ...
                  'javax.net.ssl.SSLSocketFactory');
props.setProperty('mail.smtp.socketFactory.port','465');
emailto = 'pratyushghosh1996.pg@gmail.com';
sendmail(emailto, 'MATLAB message', 'video is processed');
```

Figure 101 The Discrete Wavelet Transform based Watermarking Code (Part 5)

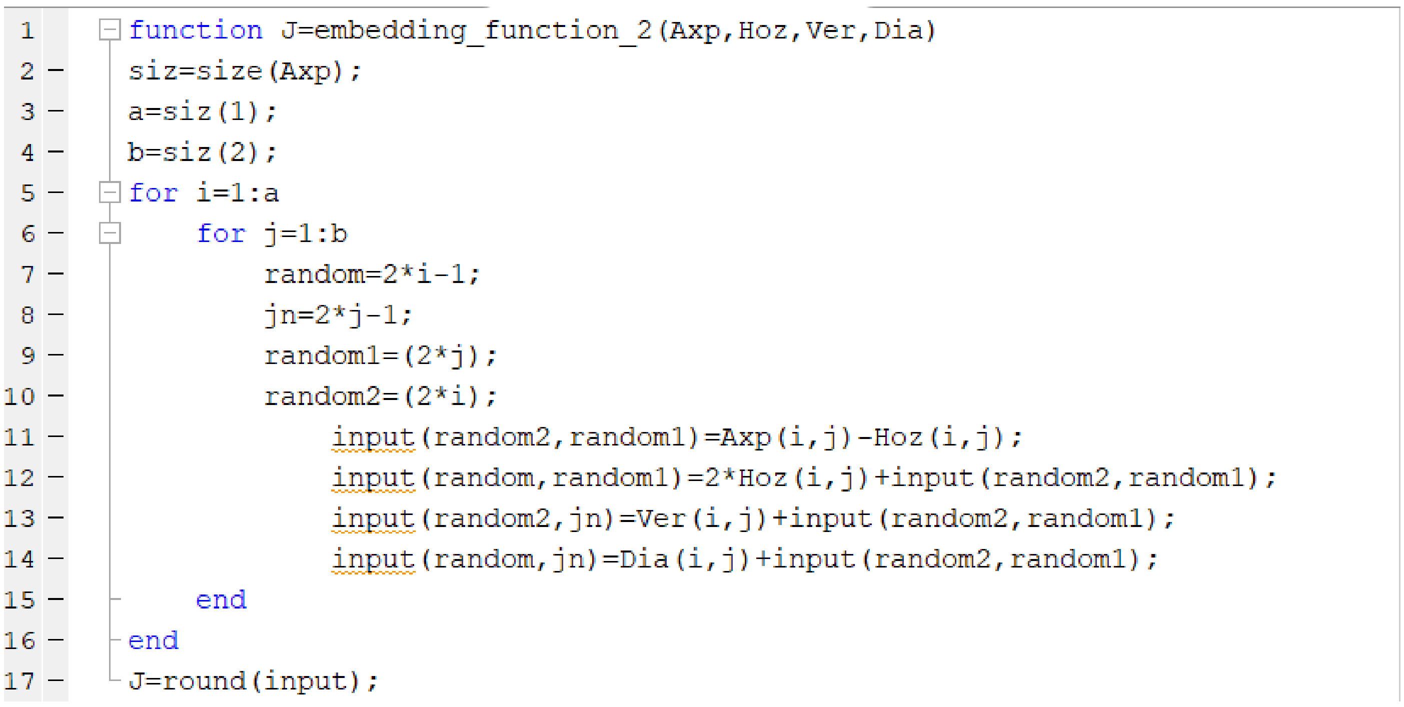
a) ***The Embedding Function 1***



```
function [App, Hoz, Ver, Dia]=embedding_function_1(im)
sz1=size(im);
a=fix(sz1(1)/2);
b=fix(sz1(2)/2);
App=zeros(a,b);
Hoz=zeros(a,b);
Ver=zeros(a,b);
Dia=zeros(a,b);
for i=1:a
    for j=1:b
        in=2*i-1;
        jn=2*j-1;
        jn1=2*j;
        in1=2*i;
        if in<=sz1(1)&& in1<=sz1(1)&& jn<=sz1(2)&& jn1<=sz1(2)
        dum1=im(in1,jn1)+im(in,jn1);
        LL1(i,j)=(round(dum1./2));
        LH1(i,j)=(LL1(i,j)-im(in1,jn1));
        HL1(i,j)=(im(in1,jn)-im(in1,jn1));
        HH1(i,j)=(im(in,jn)-im(in1,jn1));
        end
    end
end
App=LL1;
Hoz=LH1;
Ver=HL1;
Dia=HH1;
```

Figure 102 Embedding Function 1 Code

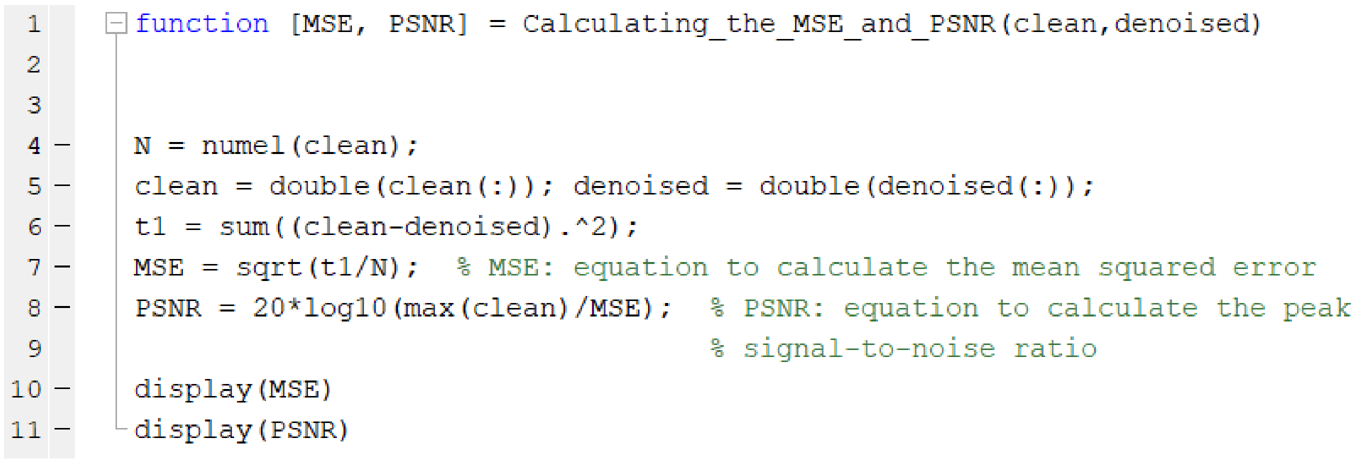
b) ***The Embedding Function 2***



```
function J=embedding_function_2(Axp,Hoz,Ver,Dia)
siz=size(Axp);
a=siz(1);
b=siz(2);
for i=1:a
    for j=1:b
        random=2*i-1;
        jn=2*j-1;
        random1=(2*j);
        random2=(2*i);
            input(random2,random1)=Axp(i,j)-Hoz(i,j);
            input(random,random1)=2*Hoz(i,j)+input(random2,random1);
            input(random2,jn)=Ver(i,j)+input(random2,random1);
            input(random,jn)=Dia(i,j)+input(random2,random1);
    end
end
J=round(input);
```

Figure 103 Embedding Function 2 Code

c) ***Calculating the MSE and PSNR Function***



```
function [MSE, PSNR] = Calculating_the_MSE_and_PSNR(clean,denoised)


N = numel(clean);
clean = double(clean(:)); denoised = double(denoised(:));
t1 = sum((clean-denoised).^2);
MSE = sqrt(t1/N);  % MSE: equation to calculate the mean squared error
PSNR = 20*log10(max(clean)/MSE);  % PSNR: equation to calculate the peak
                                  % signal-to-noise ratio
display(MSE)
display(PSNR)
```

Figure 104 The Code for calculating the MSE and PSNR values