



MIDDLE EAST TECHNICAL UNIVERSITY

Electrical & Electronics Engineering

EE 300 Summer Practice Report

Student Name: Mustafa Sefer DESTEGÜL

Student ID: 2109973

Summer Practice Date : 19.07.2019- 15.08.2019

Company: ASISGUARD

Division : Embedded Systems Department

Location of the Company: Metu technopark MET
Campus. METU, Çankaya, Ankara.

Responsible Engineer in Summer Practice: Akın
GUNONU

Email:

TABLE OF CONTENTS

1. INTRODUCTION.....	3
2. DESCRIPTION OF THE COMPANY	3
2.1. COMPANY NAME	3
2.2. COMPANY LOCATION.....	4
2.3. GENERAL DESCRIPTION OF THE COMPANY.....	4
3. INTRODUCTION TO MATLAB.....	4
4. INTRODUCTION TO VIDEO DENOISING.....	8
5. NOISE MODELS	8
5.1. GAUSSIAN NOISE	
5.2. SALT & PEPPER NOISE	
5.3. SPECKLE NOISE	
6. SOME FILTERS USED IN VIDEO PROCESSING	10
6.1. WIENER FILTER	
6.2. TEMPORAL FILTER	
6.3. MEDIAN FILTER	
7. IMAGE QUALITY MEASURING TOOLS.....	16
8. NOISY IMAGE VARIANCE ESTIMATION.....	17
9. FIRST PROJECT	22
10. SECOND PROJECT	25
11. COMPARISON BETWEEN MEDIAN FILTER AND SPATIO-TEMPORAL FILTER.....	33
12. CONCLUSION.....	37
13. REFERENCES	38

1. INTRODUCTION

I have performed my summer practice in ASISGUARD. The company has a very modern solutions in defence industry and developing fast and safe products to prevent the external dependence in the sector. My internship lasted 20 days. Akin GÜNÖNÜ , R&D group manager in ASISGUARD, was my supervisor and he arranged my internship program. I preferred ASISGUARD for my SP since the company is heavily working on the field of what I want to work in the future, embedded software and hardware designs. During my internship , I learned many things about electronics engineering and observed the facilities of the company. Moreover, I knew that ASISGUARD is following the advancements in electronics closely to make more modern and stable electronics and software designs. After starting my internship, I was charged with some projects and I completed them. The projects are mainly about the video processing tasks which will be implemented on embedded platforms such as FPGAs and embedded development boards. Since the internship program has a short period of time to implement this tasks on real-time hardware, I implemented them by the help of MATLAB. First week, I studied on MATLAB to learn more about video processing tools and functions about noise reduction on videos. Second week ,I studied how to denoise the noise on thermal videos. I learned that there are possible two types noise on thermal images , Spatial and Temporal noises. For the spatial noises the wiener filter gives a well results to eliminate the noise and also for the temporal noises, Motion compensation method is a well known to work on. Third week,as first Project, I wrote a script that will denoise the Spatial noises on thermal images. As second Project I was asked how to increase the performance of the Project. I searched and find out that some constant values , used in thresholding and weighted sum of the frames, can be made adaptive according to the frames streaming and also adding temporal filter. Furthermore, I improved my system performance by making the wiener filter and recursive temporal filter as adaptive to the noisy images' noise variance . After that, I compared my results with the pre-built median filter in MATLAB with the consideration of SSI(Structural Similarity Index) , PSNR(peak Signal to Noise Ratio) and MSE(Mean Squared Error).

In this report, I start with introducing the company. After that, I reported what I have performed and learned during SP. I have given detailed information about what I have done in projects: codes, results, performance rate, etc. At this part, I used some figures and references to explain better my works. After that, I summarized my report in "Conclusion" part. I included the sources of the document from which I took help in the "References" part.

2. DESCRIPTION OF THE COMPANY

2.1. COMPANY NAME

ASISGUARD

2.2 COMPANY LOCATION

Adress-1 : İstanbul Üniversitesi Avcılar Yerleşkesi, Üniversite Mahallesi. Sarıgül sokak. 37/1 A Block 34320 Avcılar/ İstanbul TR.

Phone : +90 212 855 54 84

URL Adress: <https://www.asiselektronik.com.tr/>

Adress-2 : Mustafa Kemal Mahallesi. Dumlupınar Bulvarı. ODTU MET BIM. No:280 A block and 4th floor. Çankaya/Ankara.

Phone : +90 312 219 00 43

URL Address: [https:// www.asisguard.com.tr](https://www.asisguard.com.tr)

2.3 GENERAL DESCRIPTION OF THE COMPANY

Asis Electronic and Information was established in 2007 to develop electronic and information system solutions within the smart city concept. In 2008, with the knowledge and the experience gained from the previous projects within Turkey and abroad, in accordance with the vision to develop national system solutions needed by our country and allied countries the re-organization started and as fo 2019 defence indusrty activities were maintained under the trademark ASISGUARD, by investing it own capital intellectual. The objects of the ASISGUARD 2019 strategic plan is to expand the product portfolio and to extend export sales. In this context, CMMI(Capability Maturity Model Integration) Level 3 certification was added to the existing quality system certifications.ASİSGUARD currently develops defence systems and Subsystems/ Hardware and Software in the fiels of Military Vehicle Electronics(Vetronics), Autonomus Micro, Mini and Medium Class Unmanned Aerial Vehicles, A4ISR,Electro-Optics,Border Security, Artificial Intelligence, Cyber Security and Big Data.

ASİSGUARD aims to enter the inventories of the Turkish Armed Forces, Law Enforcement and other allied countries witjh the systems which is researched, develop and manufactured with its own intellectuality, some of which are the firsts in our country. ASİSGUARD is a Turkish company that is taking firm steps with inn genuity towards being a main role actor in the defence industry by integrating potential and capabilities, with reliable and sustainable systems, at high technology focused ecosystem.

3. INTRODUCTION TO MATLAB

Matlab is a very powerful engineering tool at every level. Lots of things are simulated in MATLAB before the production. Thus, it is vital to be able to use MATLAB at some level. MATLAB also provides a very fast confirmation for engineers before going the hardware implementation to verify the functionality of the software. I used matlab to verify the algorithms to be implemented in hardware to be sure if they give desired outputs or not. Before my internship, I only used matlab

in some courses for some specific projects in METU EE department. Therefore in my first week of internship I wanted to make a gentle introduction to MATLAB for video processing purposes since I don't have a deep understanding of the usage of video processing in MATLAB.

3.1 dir() Function;

dir() function basically lists files and folders in the current folder that match. For example, dir(**name**) lists files and folders that match **name**. When name is a folder, dir lists the contents of the folder. I used this function in my script to take images in current folder to the workspace since I implement my denoising functions in sample images.

3.2 rgb2gray() Function;

rgb2gray(A) function converts the truecolor image A to the grayscale intensity image. The rgb2gray function converts RGB images to grayscale by eliminating the hue and saturation information while retaining the luminance. I used this function in my project to convert RGB truecolor images to grayscale images since grayscale is much more convenient color space in video processing than RGB because of the CPU performance constraints and less computing time.

3.3 Imnoise() Function;

imnoise() function provides a variety of choices to add some noise to a grayscale image like gaussian, poisson, salt&pepper and speckle noises. This function has a very wide options to add lots of different type noises to images, Furthermore; we can choose the mean and variance of the noise to be added to the grayscale image and even we can add gaussian White noise having local variance. Since I am not working in a real-time processing platform I needed to create a image with different type of noises to denoise later.

In addition to these pre-built functions, I needed to create sum functions to make the process more clear to understand.

3.4 wienerfiltercalculation(g,M) ;

This function takes an grayscale noised image and mask size as input arguments and apply some algorithms explained in 8.1 section to apply wiener filter to decrease the spatial noise on the image. This method will be explained in 8.1 section for detailed information. The MATLAB repository is taken as reference and manipulated a little for our purposes in Second Project. The MATLAB code block can be found in the figure 1.

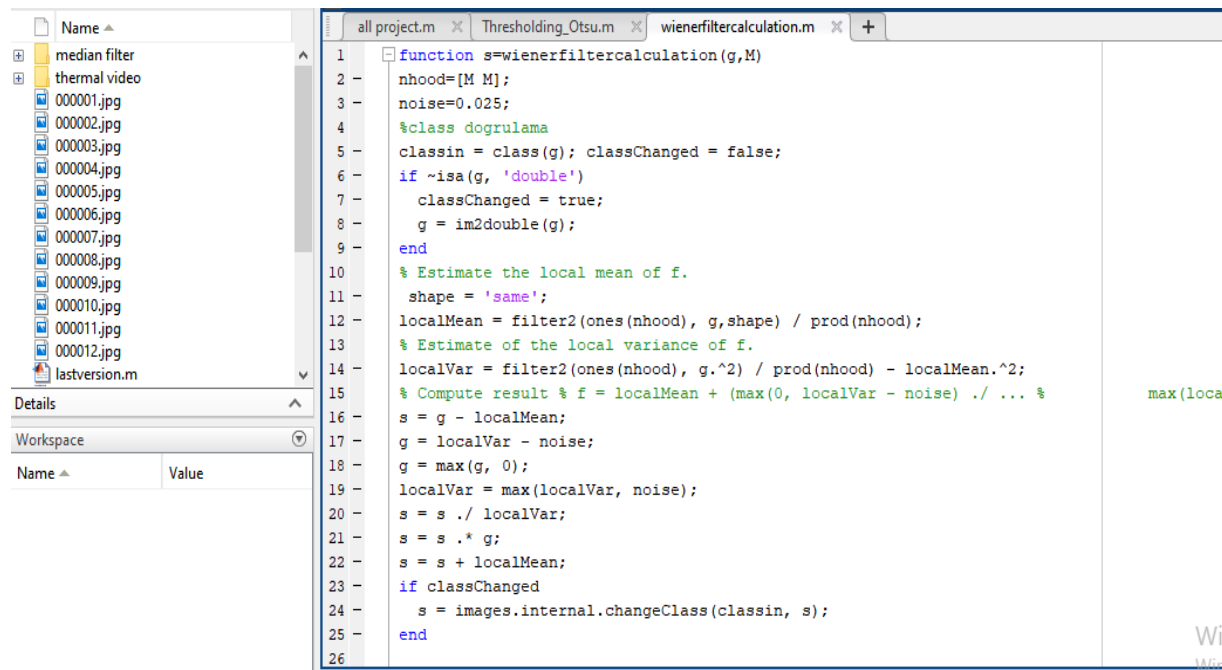


Figure 1:Wiener Filter calculation MATLAB code block

3.5 Thresholding_Otsu(image)

In computer vision and image processing, Otsu's method, named after Nobuyuki Otsu, is used to perform automatic image thresholding. In the simplest form, the algorithm returns a single intensity threshold that separate pixels into two classes, foreground and background [1]. This threshold is determined by minimizing intra-class intensity variance, or equivalently, by maximizing inter-class variance. Otsu's method is one-dimensional discrete analog of Fisher's Discriminant Analysis is a related to Jenks Optimization method, and is equivalent to a globally optimal performed on the intensity histogram. The algorithm exhaustively searches for the threshold that minimizes intra-class variance, defined as a weighted sum of variance of two classes:

$$\sigma_w^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t)$$

,weights ω_0 and ω_1 are the probabilities of the two classes separated by a threshold t , and σ_0^2 and σ_1^2 are variances of these two classes.

The class probability ω_0 and ω_1 is computed from the L bins of the histogram :

$$\omega_0(t) = \sum_{i=0}^{t-1} p(i)$$

$$\omega_1(t) = \sum_{i=t}^{L-1} p(i)$$

For 2 classes, minimizing the intra-class variance is equivalent to maximizing the inter-class variance:

$$\begin{aligned}\sigma_b^2(t) &= \sigma^2 - \sigma_w^2(t) = \omega_0(\mu_0 - \mu_T)^2 + \omega_1(\mu_1 - \mu_T)^2 \\ &= \omega_0(t)\omega_1(t)[\mu_0(t) - \mu_1(t)]^2\end{aligned}$$

Which is expressed in terms of class probabilities w and class means u where the class means $u_0(t)$, $u_1(t)$ and u_T are:

$$\mu_0(t) = \frac{\sum_{i=0}^{t-1} ip(i)}{\omega_0(t)}$$

$$\mu_1(t) = \frac{\sum_{i=t}^{L-1} ip(i)}{\omega_1(t)}$$

$$\mu_T = \sum_{i=0}^{L-1} ip(i)$$

The following relations can be easily verified:

$$\begin{aligned}\omega_0\mu_0 + \omega_1\mu_1 &= \mu_T \\ \omega_0 + \omega_1 &= 1\end{aligned}$$

The class probabilities and class means can be computed iteratively. This idea yields an effective algorithm.

In the Figure 2 MATLAB code the Otsu's method can be seen clearly.

```

1 function [threshold_otsu] = Thresholding_Otsu(Image)
2
3
4     nbins = 256;
5     counts = imhist(Image,nbins);
6     p = counts / sum(counts);
7
8     for t = 1 : nbins
9         q_L = sum(p(1 : t));
10        q_H = sum(p(t + 1 : end));
11        miu_L = sum(p(1 : t) .* (1 : t)) / q_L;
12        miu_H = sum(p(t + 1 : end) .* (t + 1 : nbins)) / q_H;
13        sigma_b(t) = q_L * q_H * (miu_L - miu_H)^2;
14    end
15
16    [~,threshold_otsu] = max(sigma_b(:));
17 end

```

Figure 2: Otsu's Thresholding Method MATLAB code block

4. INTRODUCTION TO VIDEO DENOISING

During the first week of my internship, I studied the basics of video denoising. After some researches, I found out the foremost factor which influences the quality of thermal images is presence of noise. It is common that the signal-to-noise ratio is low and the contrasts of thermal images is low as well. These two reasons make the processing of thermal image difficult. The sources of a big level of noise are IR sensors and the interference of the signal processing circuit. Depending on the IR sensor technology it may be the problem connected with photoelectric conversion (in photodetectors) or temperature fluctuation noise (in thermal detectors) as well as the influence of manufacturing process [2]. The problems mentioned above cause IR detector non-uniformity, which causes varied response of the pixels. This non-uniformity may be the main source of the noise in thermal images if not properly corrected. Furthermore, I learned that the environmental conditions during the measurement has to be considered.

There are lots of methods of noise removal. Some of them were adopted from traditional image processing, like for example the Wiener filter; some others were converted to improve their influence on thermal images like the center-weighted median filter and methods related to fuzzy logic. The classical methods of noise removal, for example the Median filter, can be used in the case of thermal images, but the results can be very blurred. The reason behind it is the fact that this type of filters has been dedicated to visual image processing, where the signal-to-noise ratio is much more bigger than one for thermal images. After using the median filter, the important details and small elements can be lost due to the fact that median filter has influence on every pixel of image both the disturbed and the undisturbed. Therefore another method has to be taken into consideration.

5. NOISE TYPES

Noise tells unwanted information in digital images. Noise in a video frame produces undesirable effects such as artifacts, unrealistic edges, unseen lines, corners, blurred objects and disturbs background scenes. To reduce this unwanted effects, it is essential to learn about the noise models for further processing [3].

5.1. GAUSSIAN NOISE

The random noise that enters the system can be modelled as Gaussian or Normal distribution. The Gaussian distribution is a well known bell-shaped curve as can be seen from the figure 3. This is mathematically denoted as $F = S \pm N$, where N is the Gaussian probability density function (PDF) and S is the noiseless image. The Gaussian noise affects both the dark and light areas of an image. The Gaussian distribution can be seen from Figure 3.

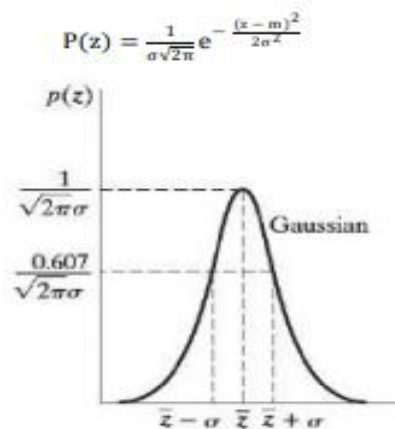


Figure 3: Gaussian Distribution

The output for the Gaussian noise can be seen from figure 4.



Figure 4: a) original image b) noisy image with Gaussian noise of variance 0.03 and mean 0

5.2 . SALT & PEPPER NOISE

This is a type of noise, which is also known as impulse noise, shot noise and binary noise. It is caused by sensor and memory problems due to which pixels are assigned incorrect maximum values. The salt and pepper noisy image can be seen from the figure 5.



Figure 5: Original and salt& pepper noisy image, respectively.

5.3 SPECKLE NOISE

This noise is type of multiplicative noise. It is generally found in medical images and appears as bright specks in the lighter parts of an image.

$$I = S + (S * N_g)$$

The above mathematical notation represents speckle noise, where N_g is the random noise, which has zero-Mean Gaussian probability distributive function. The Speckle type noise can be seen from the Figure 6.

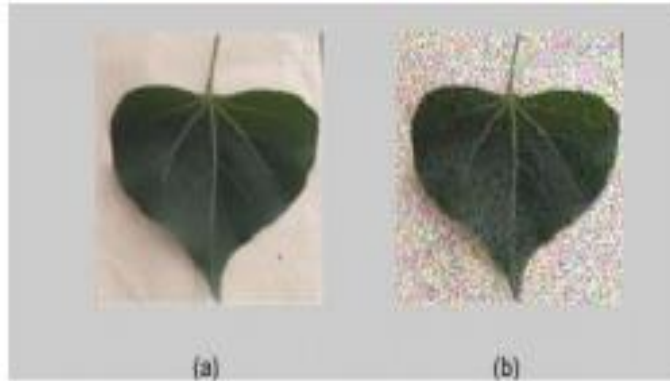


Figure 6: Original and Speckle noisy image, respectively.

6. SOME FILTERS USED IN VIDEO PROCESSING

There are many video noise reduction algorithms existing beforehand in the literature. These algorithms can be classified into three categories: first category implements in a spatial domain[4], second category implements in a temporal domain [5,6] ,while the third category implements in the combination of a spatial and temporal domain[7,8]. As an example of spatial filter, Wiener filter is considered as a classical approach for spatial noise filtering. This filter has the capacity to achieve high gain in noise removal. However, it can cause serious damage to the edge of the image during the process of the noise removing, especially in noise-free areas as mentioned before. On the other hand , Yan and Yanfeng in[9] proposes a noise reduction method based on temporal filtering. The basic idea of this method is to remove the noise in continuous frames. According to Yan and Yanfeng , the proposed filter has succeeded in reducing the noise in real video sequence. Nevertheless, this filter suffers from the dragging effects on moving objects. Although the wiener filter and temporal filtering method have good results, they have some weakness like blurring and dragging effects.

6.1. WIENER FILTER

The Wiener filtering is optimal in terms of the mean square error. In other words, it minimizes the overall mean square error in the process of inverse filtering and noise smoothing. The Wiener filtering is a linear estimation of the original image. The approach is based on a stochastic framework[10]. The orthogonality principle implies that Wiener filter in Fourier domain can be explained as follows:

$$W(f_1, f_2) = \frac{H^*(f_1, f_2)S_{xx}(f_1, f_2)}{|H(f_1, f_2)|^2 S_{xx}(f_1, f_2) + S_{\eta\eta}(f_1, f_2)},$$

Where $S_{xx}(f_1, f_2)$, $S_{\eta\eta}(f_1, f_2)$ are respectively power spectra of the original image and the additive noise, and $H(f_1, f_2)$ is the blurring filter. We can see that Wiener filter has two parts, an inverse filtering part and a noise smoothing part. It can be seen that the formula not only performs the deconvolution by inverse filtering (highpass filtering), but also removes the noise with a compression operation (lowpass filtering).

To implement the Wiener filter in practice we have to estimate the power spectra of the original image and additive noise. For white additive noise, it seems that the power spectrum is equal to the variance of the noise while the power spectrum of the original image can be found by various methods. According to [3] the direct estimate is the periodogram estimate of the power spectrum computed from the observation:

$$S_{yy}^{per} = \frac{1}{N^2} [Y(k, l)Y(k, l)^*]$$

where $Y(k, l)$ is the DFT of the observation. We can make use of this equation without worrying about the singularity of the inverse filtering. The other estimation which leads to a cascade implementation of the inverse filtering and the noise smoothing is:

$$S_{xx} = \frac{S_{yy} - S_{\eta\eta}}{|H|^2},$$

which is the direct result of the fact $S_{yy} = S_{\eta\eta} + S_{xx}|H|^2$. The periodogram estimation gives us the power spectrum S_{yy} directly from the observation. This results in a cascade implementation of inverse filtering and noise smoothing:

$$W = \frac{1}{H} \frac{S_{yy}^{per} - S_{\eta\eta}}{S_{yy}^{per}}.$$

The MATLAB code for Wiener filter calculation can be found in section 3.4.

6.2 TEMPORAL FILTER

Temporal noise is the temporal variation in pixels output values under constant illumination due to device noise, supply and substrate noise, and quantization effects. This results in a speckled noise with pixilation that looks like staticky TV screen. The reason is that there is just not enough light hitting the sensor. In bright conditions, all the light provides a huge signal ; noise- from electrical interference or imperfectib in the detector. In low light, the signals are much smaller which means that thenoise is painfully apperent. It exploits the fact that with video there are two pools of data to use: each separate image, and the knowledge of how frames change with time. Using that information, it is possible to create an algorithm that can work out which pixels havechanged between frames. But it is also possible to work put which pixels are expected to change between frames. For instance, if a car is moving from left to right in a frame, software can soon work out that pixels to the right should change dramatically[11].

For that purpose ,As a temporal filter, Motion compensation method is widely used in temporal filtering. Motion compensation is an algorithmic technique which used to prodict a frame in a video, given the previous and future frames by counting for motion of the camera and/or objects in the video. Motion Compansation describes a picture in terms of the transformation of a reference picture to the current picture. The reference picture may be previous in time or even from the future. Motion Compensation exploits the fact that , for many frames of a movie, the only difference between one frame and another is the result of either the camera moving or and object in the frame moving. In reference to a video fie, this means much of the information l that represents one frame will be the same as the information used iin the next frame. Using motion Compensation, a video stream will contain some full(reference) frame; then the only information stored for the frames in between would be the information needed to transform the previous frame into the next frame. However the current pixel and the matching pixel might be located in different coordinated when motion occurs. A local motion vector is used to align the moving parts in the image sequences, so that temporal recursive filtering can use the information correctly from frame to frame. In very noisy situations, it is hard to determine the correct motion vector. Motion detectiin is an alternative solution to motion compensation[12].

The output of the motion detection based recursive temporal filter can be written as

$$T_n(i, j) = \rho \cdot f_{n-1}(i, j) + (1 - \rho) \cdot f_n(i, j),$$

Where f_n donate the nth current frame filtered by spatian Wiener filter and T_n is the temporally filtered nth frame and ρ is;

$$\rho(i, j) = \frac{\sigma_n^2(i, j)}{\sigma_r^2(i, j) + \sigma_n^2(i, j) + \sigma_r(i, j) \cdot \sigma_n(i, j)}$$

And f_{n-1} is the previous frame filtered by spatial Wiener filter. In section 8, I will be explaining how we can find the noisy image variance given an noisy image.

The MATLAB code of the temporal filter code can be found in figure 7.

```

1 T=input('enter threshold value for recursive temporal filtering');
2 a=0.1002; %Fix any value of a, then optimize T. Next Fix T, optimize a
3 [height,width,nframes] = size(Y(1));
4 h=double(zeros(height,width));
5 for f=2:nframes % the first frames will be same.
6     current=Y(:, :, f);
7     previous=Y(:, :, f-1);
8     k=current-previous;
9     [row, col]=size(current);
10    for i=1:row
11        for j=1:col
12            if 0.2
13                abs(k(i,j)) < T % motion estimation. For large motion(>T), there is less redundancy.
14                h(i,j)=a*current(i,j)+(1-a)*previous(i,j);
15            else
16                h(i,j)=current(i,j);
17            end
18        end
19    end
20    Y(:, :, f)=h;
21 end
22 Z=Y(1).name;
23 imshow(Z(1:359,1:639));
24

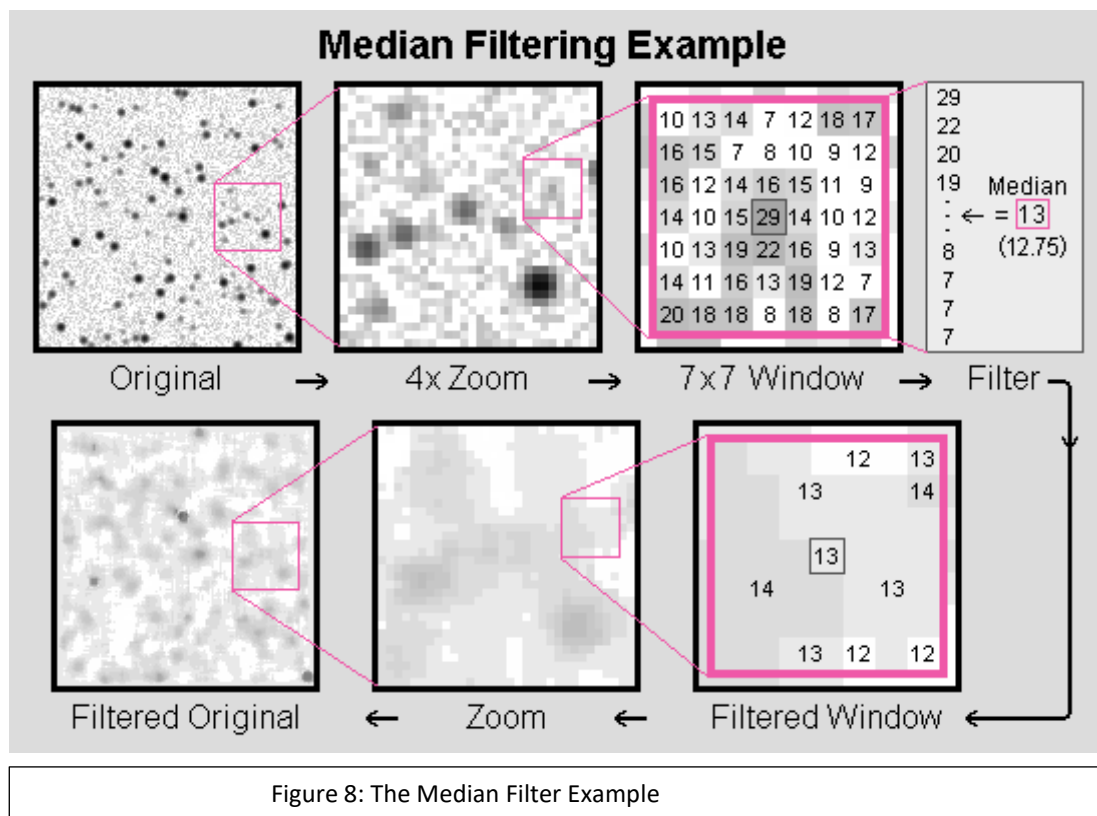
```

Figure 7: Temporal Filter MATLAB code.

6.3 MEDIAN FILTER

The median filter is a non-linear digital filtering technique, often used to remove noise from an image or signal. Such noise reduction is a typical pre-processing step to improve the results of later processing (for example, edge detection on an image). Median filtering is very widely used in digital image processing because under certain conditions, it preserves edges while removing noise

also having application in signal processing. The main idea of the median filter is to run through the signal entry by entry, replacing each entry with the median of neighboring entries. The pattern of neighbors is called “window”, which slides entry by entry over the entire signal[13]. For 1D signals, the most obvious window is just first few preceding and following entries, whereas for 2D data ,the window must include all entries within a given radius or ellipsoşdal region , which means that the median filter is not a seperable filter. This filter smooths the data while keeping the small and and sharp details. The median filter is basically just the middle value of all values of the pixels in the neighborhood. This is not the same as the average (or mean); instead, the median has half values in the neighborhood larger and half smaller. The median is a stronger “Central Indicator” than the average. In particular, the median filter is hardly affected by a small number of discrepant values among the pixels in the neighborhood. Consequently, median filtering is very effective at removing various kinds of noise. The figure 8 shows and example of median filtering.



The output of the median filter can be found in figure 9.

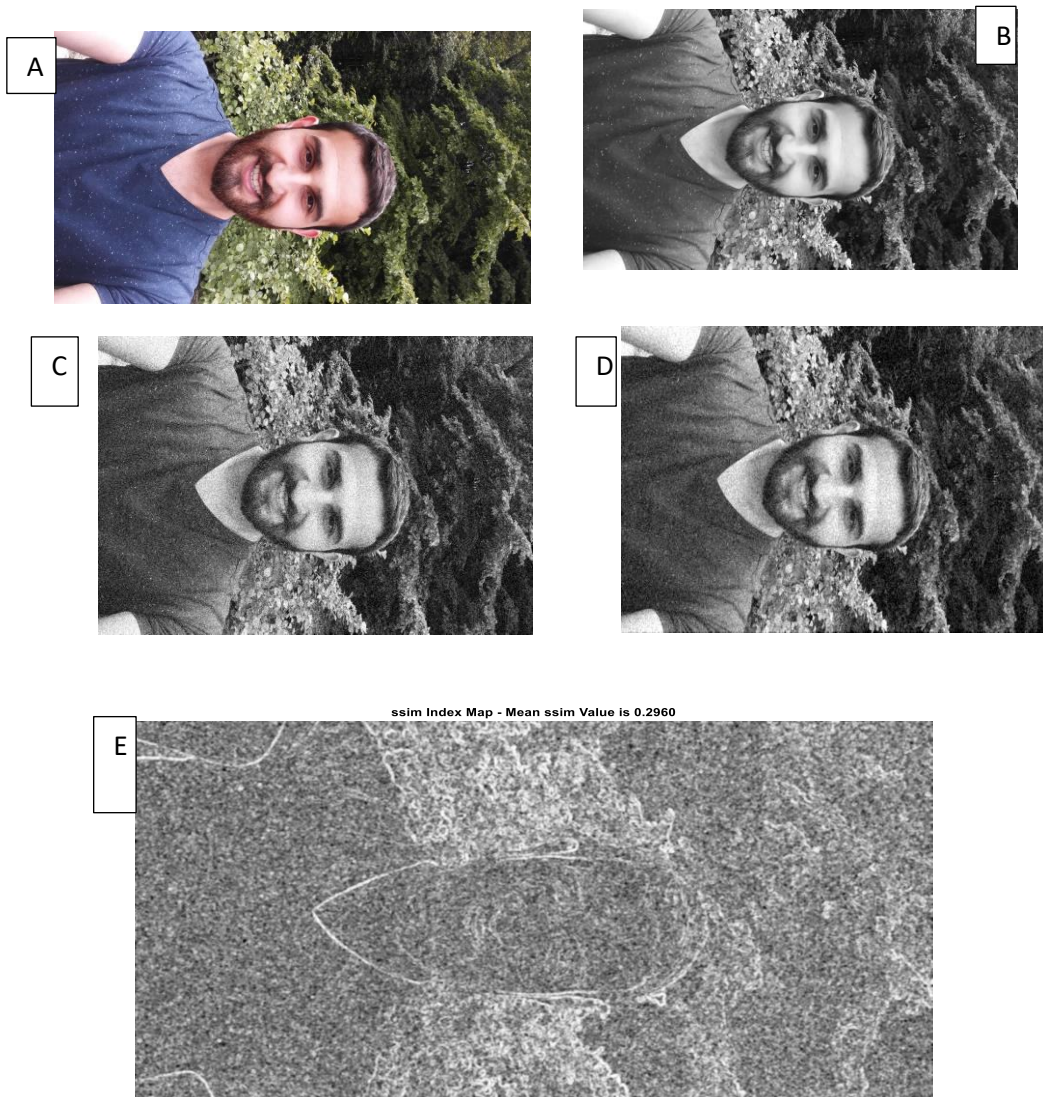


Figure 9: A: RGB image, B:Grayscale image C:Noisy image with 0.025 Gaussian Noise, D: Denoised image by Median Filter E:SSIM value(0.2960) of D compared to B

The Median filter MATLAB code is as can be seen in figure 10 and figure 11

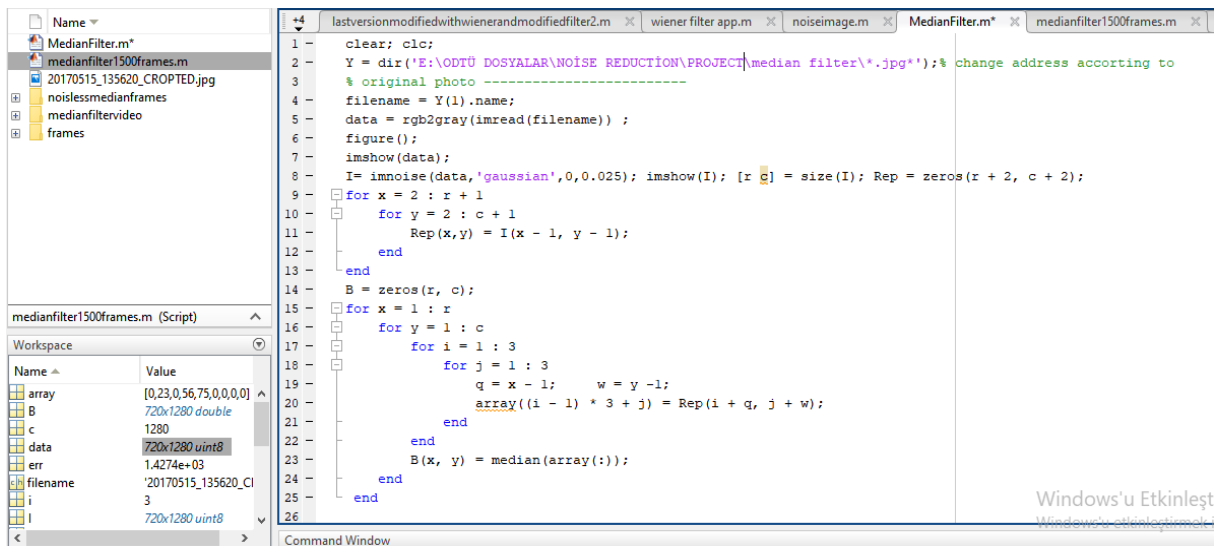


Figure 10: Median Filter MATLAB code part1

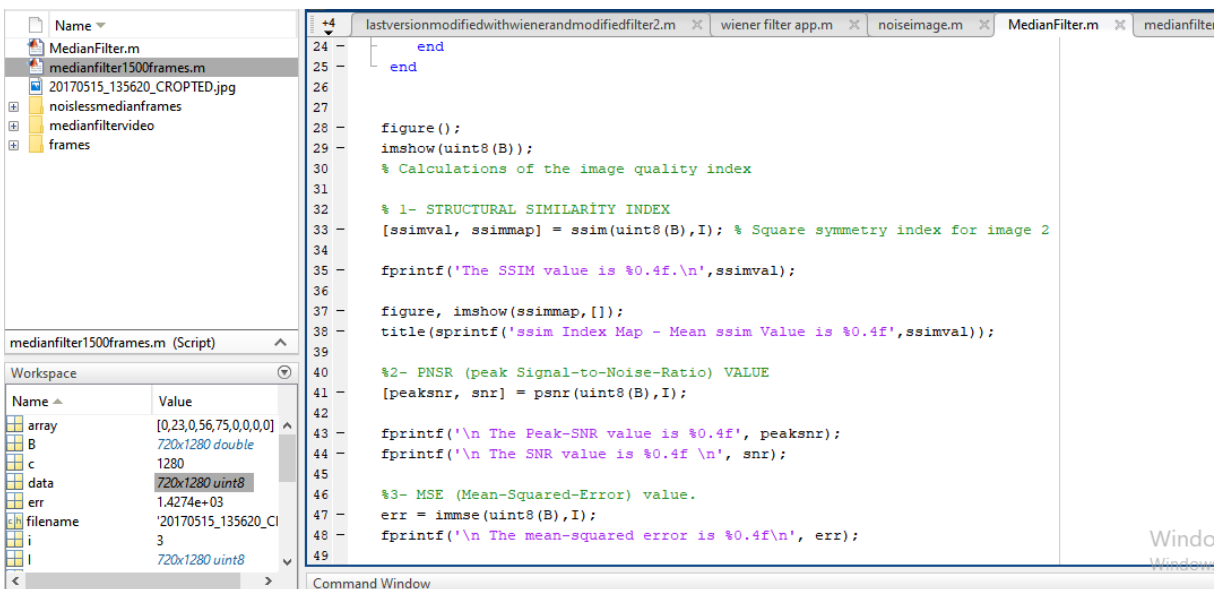


Figure 11: Median Filter MATLAB code part 2

7. PERFORMANCE MEASUREMENT TOOLS

Mean Square Error(MSE) and Peak Signal-to-noise Ratio(PSNR) are image quality measuring tools. These tools are very helpful in measuring the quality of a filtered image. The below two equations can be used to characterize the quality of an 8-bit filtered image. If these two equations are applied on a 24-bit filtered image, MSE and PSNR values are calculated for each 8-bit of the image separately[14]. As a result of which three different values of MSE and PSNR get displayed for a 24-bit image. The calculations for 1D ;

$$MSE = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} [f(x,y) - \hat{f}(x,y)]^2$$

$$PSNR = 20 \log_{10} \frac{255^2 MN}{\sum_{i=0}^{M-1} \sum_{j=0}^{N-1} [f(x,y) - \hat{f}(x,y)]^2}$$

For 2D calculations;

$$MSE = \text{sum}(\frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} [f(x,y) - \hat{f}(x,y)]^2)$$

$$PSNR = \text{sum}(20 \log_{10} \frac{255^2 MN}{\sum_{i=0}^{M-1} \sum_{j=0}^{N-1} [f(x,y) - \hat{f}(x,y)]^2}) \text{ dB}$$

Further more there is an another strong image quality performance tool which is called Structural Similarity Index(SSIM). The Structural Similarity Index (SSIM) is a perceptual metric that quantifies image quality degradation* caused by processing such as data compression or by losses in data transmission. It is a full reference metric that requires two images from the same image capture— a reference image and a processed image.

8. NOISY IMAGE VARIANCE ESTIMATION

One of the most important consideration in digital processing of images is noise, in fact it is usually the factor that determines the success or failure if any of the enhancement or reconstruction scheme, most of which in the present of significant noise. In all processing systems must consider how much of the detected signal can be regarded as true and how much is associated with the random background events resulting from either the detection or transmission process. These random events are classified under the general topic of noise. This noise can result from a vast variety of sources, including the discrete nature of radiation, variation in the detector sensitivity, photo-graphic grain effects, data transmission errors. In each case the properties of the noise are different, as are the image processing operations that can be applied to reduce their effects. For detection of the noisy pixels, I chose the Gaussian Approximation. The Central Limit Theorem for large expectation values a poisson distribution is well approximated by a Gaussian distribution of mean and variance equal to the expectation value.

$$p(n) = \frac{u^n \exp(-u)}{n!} \rightarrow \frac{1}{(2\pi u)^{1/2}} \exp\left(-\frac{(n-u)^2}{2u}\right)$$

For large u . The comparison between the Gaussian approximation and the Poisson distribution for $u=20$ is shown in figure 12 and shows this approximation is valid to within about %1 for values $u>20$.

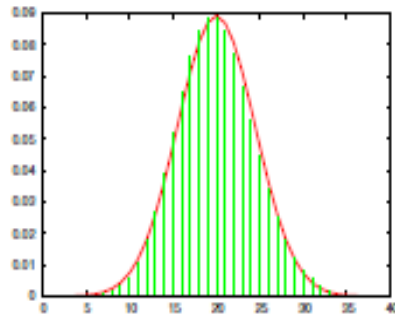


Figure 12: Gaussian Distribution for mean of 20

In most imaging systems, with the important exception for low light level astronomical images, the expected number of particles per pixel can be regarded as large, typically 1000s. We can therefore approximate the probability distribution for each image pixel by

$$p(f) = \frac{1}{(2\pi \langle f \rangle)^{1/2}} \exp\left(-\frac{(f - \langle f \rangle)^2}{2\langle f \rangle}\right)$$

The detected value f can now be treated as a true signal $\langle f \rangle$, which is characteristic of the source and a deviation of noise n , characteristic of the detection process, so that for a particular pixel ,

$$f = \langle f \rangle + n$$

Where n is a random variable with the probability distribution

$$p(n) = \frac{1}{(2\pi \langle f \rangle)^{1/2}} \exp\left(-\frac{n^2}{2\langle f \rangle}\right)$$

Therefore for a two dimensional image we have the model that the detected image $f(i,j)$ is given by

$$f(i,j) = (f)(i,j) + n(i,j)$$

Where each additive noise term is characterized by a probability distribution function with mean and variance given by the expectation value of the signal at that pixel, i.e

$$p(\pi(i, j)) = \frac{1}{(2\pi(f)(i, j))^{1/2}} \exp\left(\frac{-\pi(i, j)^2}{2(f)(i, j)}\right)$$

So that the noise term $n(i,j)$ is dependent on the signal.

The MATLAB CODE for estimating the noise variance in a noisy image is as can be seen from the figure 13, figure 14 and figure 15.

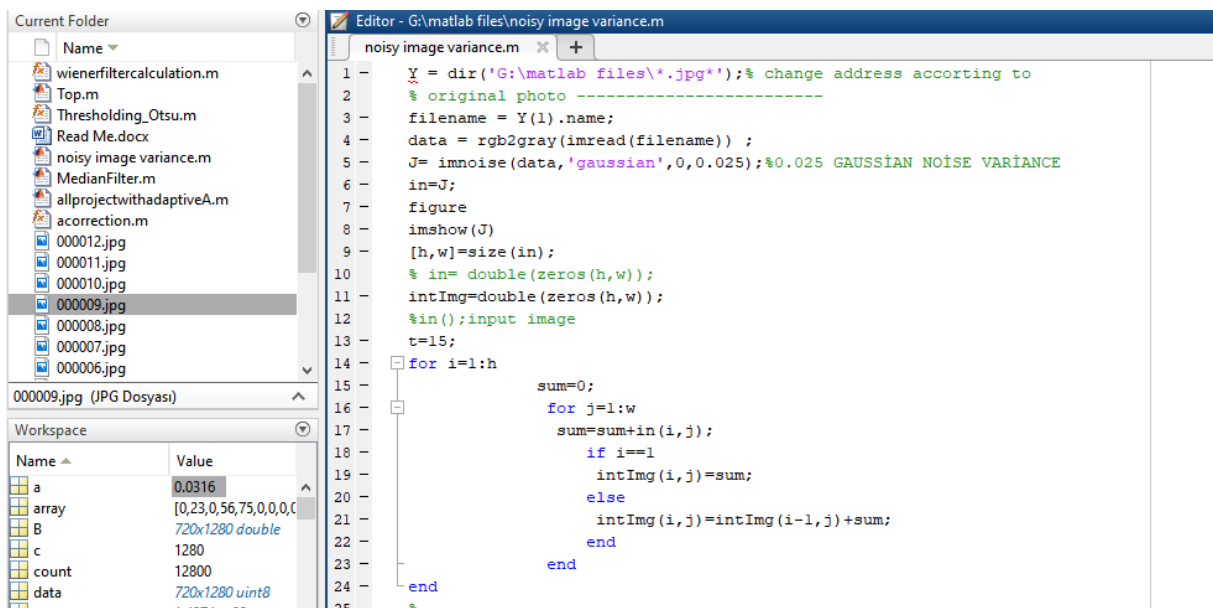


Figure 13: Noise variance Estimation for noisy images part1

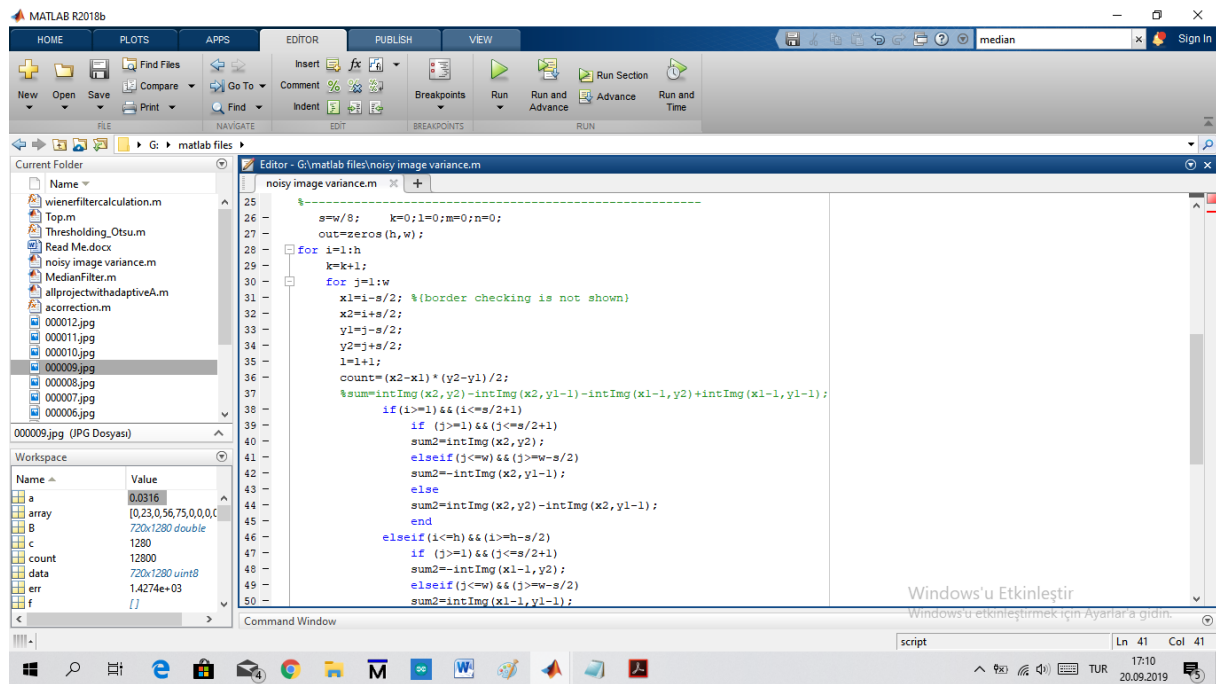


Figure 14: Noise variance Estimation for noisy images part2

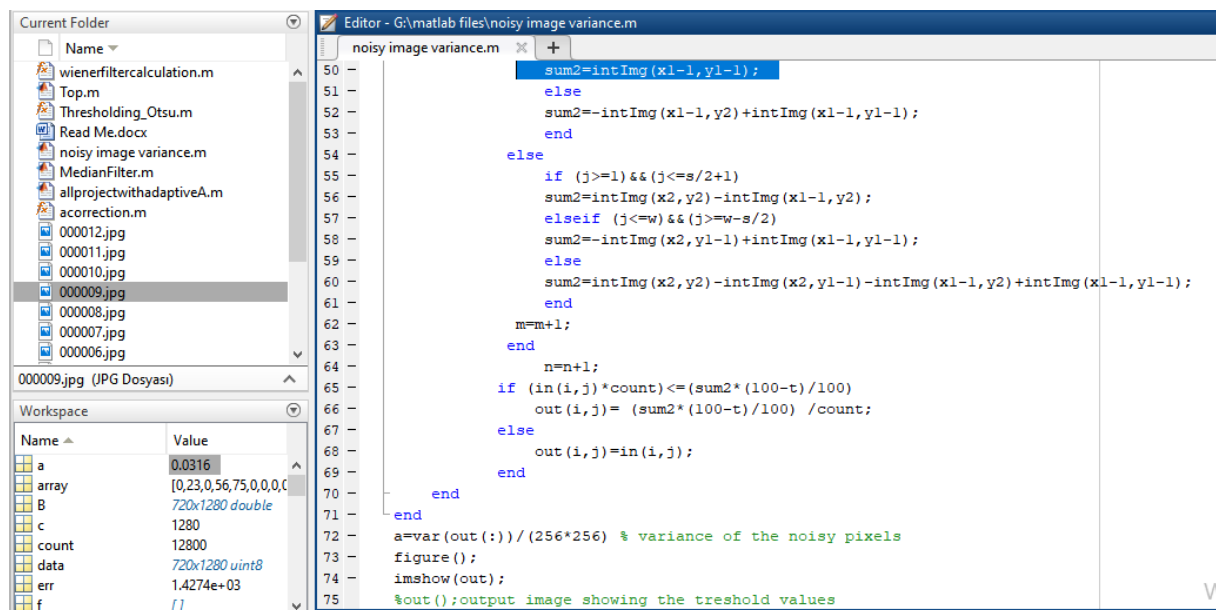


Figure 15: Noise variance Estimation for noisy images part3

The results for the Noise Estimation algorithm can be found in figure 15.1, figure 15.2 and figure 15.3.

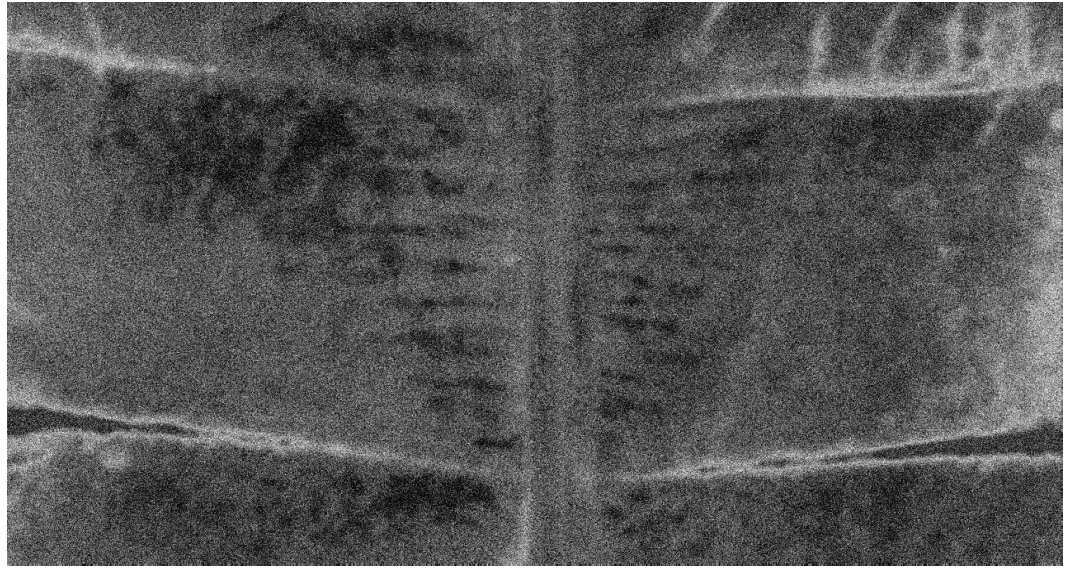


Figure 15.1: The noisy image with 0.025 Gaussian type noise.

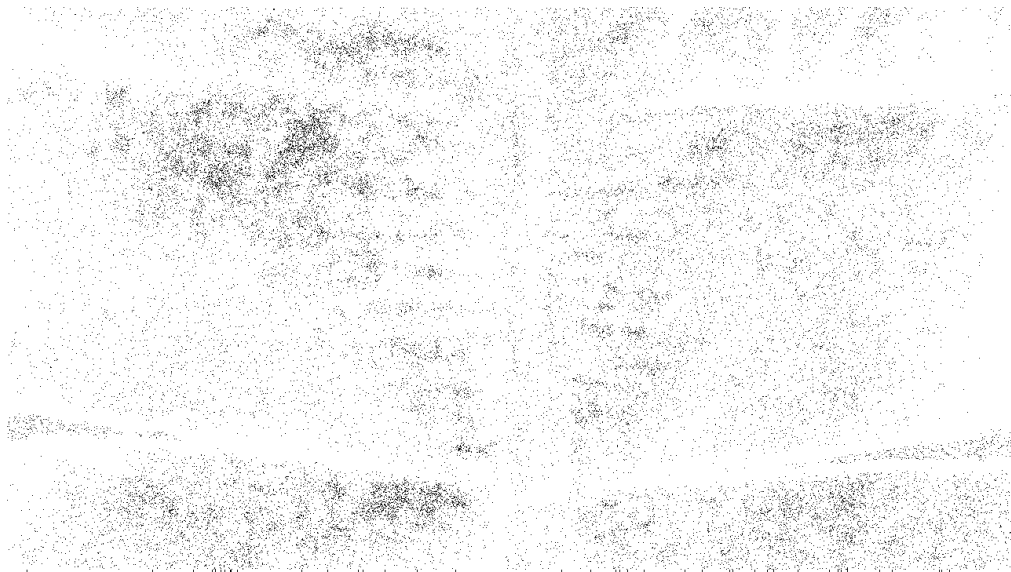


Figure 15.2: The Thresholded image showing the estimated noisy pixels

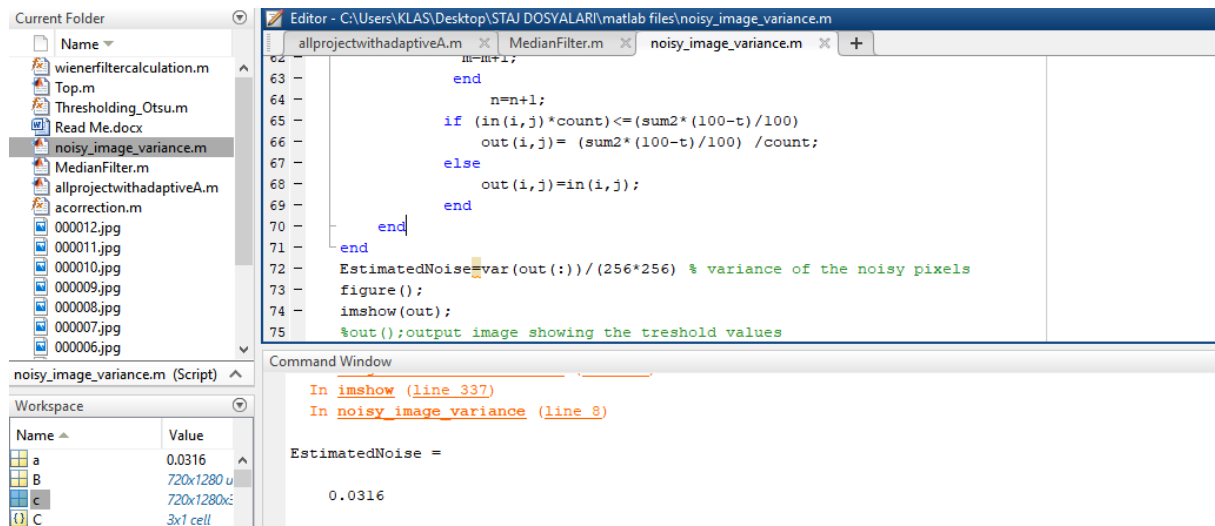


Figure 15.3: The estimated noise(0.0316)

9. FIRST PROJECT

In signal processing, the wiener filter is a filter used to produce an estimate of a desired or target random process by linear time-invariant filtering of an observed noisy process, assuming known stationary signal and noise spectra, and additive noise. The Wiener filter minimizes the mean square error between the estimated random process and the desired process. Let's suppose that $y(i, j)$ is the input corrupted with a White Gaussian noise $n(i, j)$ with zero mean and variance $\sigma_n^2(i, j)$. The observed noisy image $y(i, j)$ is known as the sum of the original $x(i, j)$ and some noise $n(i, j)$, that is, $y(i, j) = x(i, j) + n(i, j)$.

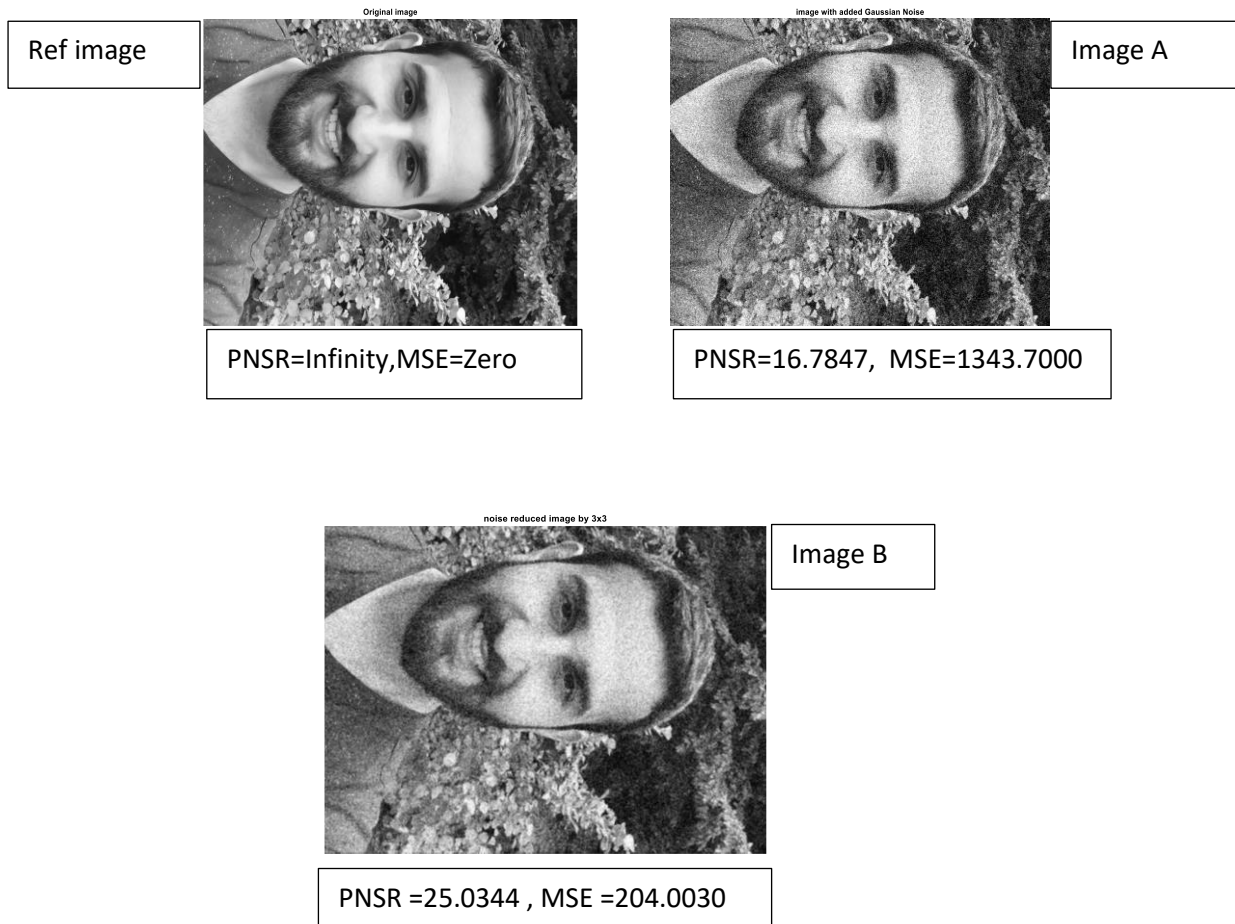
The main purpose of the noise reduction algorithms is to restore the image $R(i, j)$ from the degraded image $y(i, j)$ of the original image $x(i, j)$. The most efficient algorithm is that one which has the ability to yield image $R(i, j)$ so as to be as close as possible to the original image $x(i, j)$. Wiener filter is based on this principle.

The Wiener filter formulation can be written as

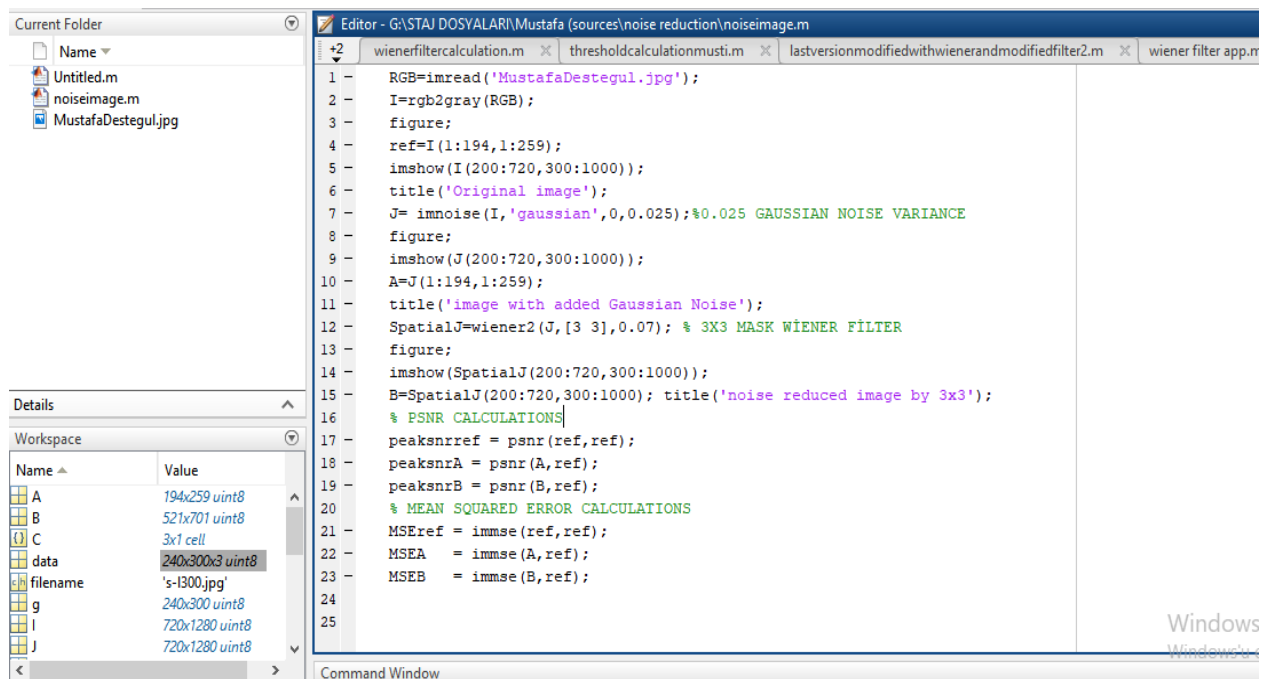
$$W(i, j) = \frac{\sigma_n^2(i, j)}{\sigma_n^2(i, j) + \sigma_y^2(i, j)} \cdot y(i, j),$$

Where $\sigma_y^2(i, j)$ is the variance of the noise over the input image (noisy image) $y(i, j)$.

The Results of Wiener Filter can be seen in Ref image, Image A and Image B which has the same name with the code.



The matlab code for the Wiener filter calculations is given in Figure 16;



```

1 - RGB=imread('MustafaDestegul.jpg');
2 - I=rgb2gray(RGB);
3 - figure;
4 - ref=I(1:194,1:259);
5 - imshow(I(200:720,300:1000));
6 - title('Original image');
7 - J= imnoise(I,'gaussian',0,0.025);%0.025 GAUSSIAN NOISE VARIANCE
8 - figure;
9 - imshow(J(200:720,300:1000));
10 - A=J(1:194,1:259);
11 - title('image with added Gaussian Noise');
12 - SpatialJ=wiener2(J,[3 3],0.07); % 3X3 MASK WIENER FILTER
13 - figure;
14 - imshow(SpatialJ(200:720,300:1000));
15 - B=SpatialJ(200:720,300:1000); title('noise reduced image by 3x3');
16 - % PSNR CALCULATIONS
17 - peaksnrref = psnr(ref,ref);
18 - peaksnrA = psnr(A,ref);
19 - peaksnrB = psnr(B,ref);
20 - % MEAN SQUARED ERROR CALCULATIONS
21 - MSEref = immse(ref,ref);
22 - MSEA = immse(A,ref);
23 - MSEB = immse(B,ref);
24
25

```

Name	Value
A	194x259 uint8
B	521x701 uint8
C	3x1 cell
data	240x300x3 uint8
filename	's-1300.jpg'
g	240x300 uint8
I	720x1280 uint8
J	720x1280 uint8

Figure 16: Wiener filter MATLAB code.

As can be seen from the ImageB, the Wiener filter has the capacity to achieve high gain in noise removal. However, it causes a very highly blurred image since the mask applied to pixels are 3x3. We can make this statement more clearer by applying a 7x7 mask to the noisy image.



Figure 17: A: 3x3 Wiener Filter

B: 7x7 Wiener filter.

As can be seen from the Figure 17, as we increased the mask size, the noise is removed more, but the image gets more blurrier. Although this method is presenting a nice noise removal, it has some problem like blurring. The reason is that we apply wiener filter denoising method on the whole frame for both low noisy areas and high noisy areas, which result in an image which has the same blur in the whole.

Furthermore, noises in an image is not only Spatial but also can be Temporal. This method doesn't have any solution for the temporal noises. In Project 2, I will be explaining how we can decrease noise much more by considering both Spatial and Temporal noises with Adaptive Wiener filter and Adaptive Recursive Temporal Filter according to the noise rate with adaptive threshold values.

10. SECOND PROJECT

In this Project I will be applying noise reduction methods on the noisy images based on the Adaptive Wiener filter and Adaptive Recursive Temporal filter. Both the amount of noise and the size of the mask will be taken into consideration. It has a good capacity to be adaptive in each area in accordance with the amount of noise. In this method, the motion detector is applied to control the noise removal process in accordance with the area's information (i.e, static or movable). More accurately, more noise removal is done in the areas that are potentially still areas and less removal in the areas that are potentially motion areas. The results demonstrate that the new approach is more efficient than the one presented in the First Project in terms of noise removal and edges preservation[16].

Frames in video sequence are temporally associated. Wherefore in video noise reduction algorithms, the temporal filter should be used in the motion areas of the frames to diminish the noise extent practicable. However, temporal filter cannot be utilized alone since it may cause blurring in the motion areas. On the other hand, utilizing the spatial filter separately often causes spatial blurring. For that the spatial filter must be used in combination with temporal filter[17].

For that purpose, the recursive time averaging will be applied in the areas where the motion has not been detected. This model is able to be adapted in each area depending on the information of the area. More precisely, The spatial-temporal recursive filter is going to play a good role in making noise removal more at the still areas and less at the motion areas. For example, the regulation of the filtering action is carried out by applying the mask size of 5x5 in the areas which contain high levels of noise and low image features. On the other hand, we apply the mask size of 3x3 in the areas which contain less noise and many image features. For controlling the filtering action, the following function will be used;

$$f_n(i, j) = \begin{cases} \mu > T1 & \text{Mask size of } 5 \times 5, \\ \text{Otherwise} & \text{Mask size of } 3 \times 3, \end{cases}$$

Where u is the amount of noise and T_1 is the adaptive threshold according to the noisy image.

As can be seen from the figure 18 ,in the first stage , I use the spatial Wiener filter to filter the previous and current degraded frames. The proposed Wiener adaptive filter has the ability to adapt and change in each area in accordance with the amount of noise. In the proposed spatial Wiener filter, the size of the mask is taken into consideration. More accurately, the mask with large size is applied in the areas with high noise levels, while the mask with small size is utilized in areas with low noise levels. In the second stage, in figure 19, we enhance the results of the spatial filter with temporal filter. At the last stage, we use the motion detector and recursive time averaging to improve the temporal filter's output. For the motion detector part the following motion field to control the process of removing noise according to the area information is chosen;

$$M(i, j) = \begin{cases} 0 & |f_n(i, j) - T_n(i, j)| < T, \\ 1 & \text{otherwise,} \end{cases}$$

,where $M(i,j)$ is the field with motion.

In case of $M(i,j)=0$, the changes over $f_n(i,j)$ and $T_n(i,j)$ at the spatial position (i,j) will be nearly zero. In other words, f_n nearly equal to T_n .

If $M(i,j)=1$ the changes over $f_n(i,j)$ and $T_n(i,j)$ at the spatial position (i,j) will be significant. So the noise removal should be stopped in this case. Otherwise, the significant information of the image in this area will be removed which will lead to a blurry image.

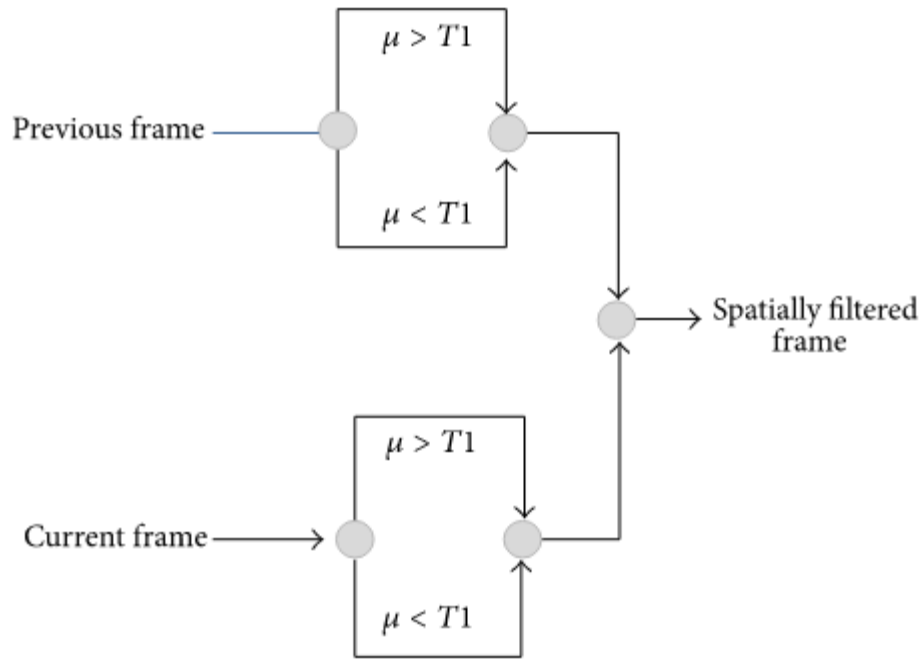


Figure 18: Adaptive Wiener Filter according to the area information.

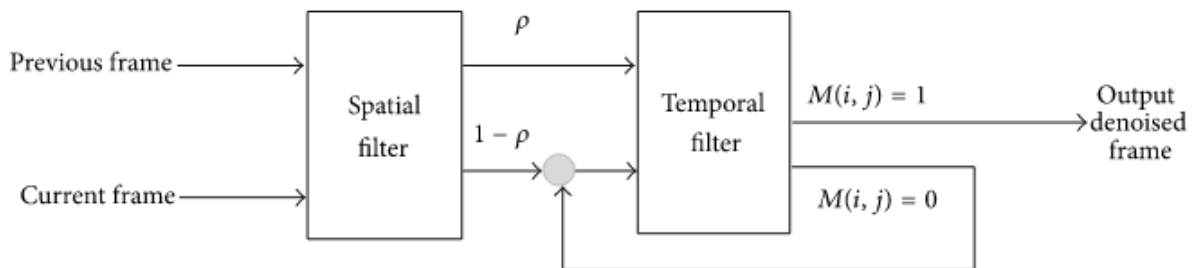


Figure 19: Spatio-Temporal Noise Reduction Filter

The matlab codes for Second Project can be seen from the figure 20, figure 21, figure 22 and figure 23;

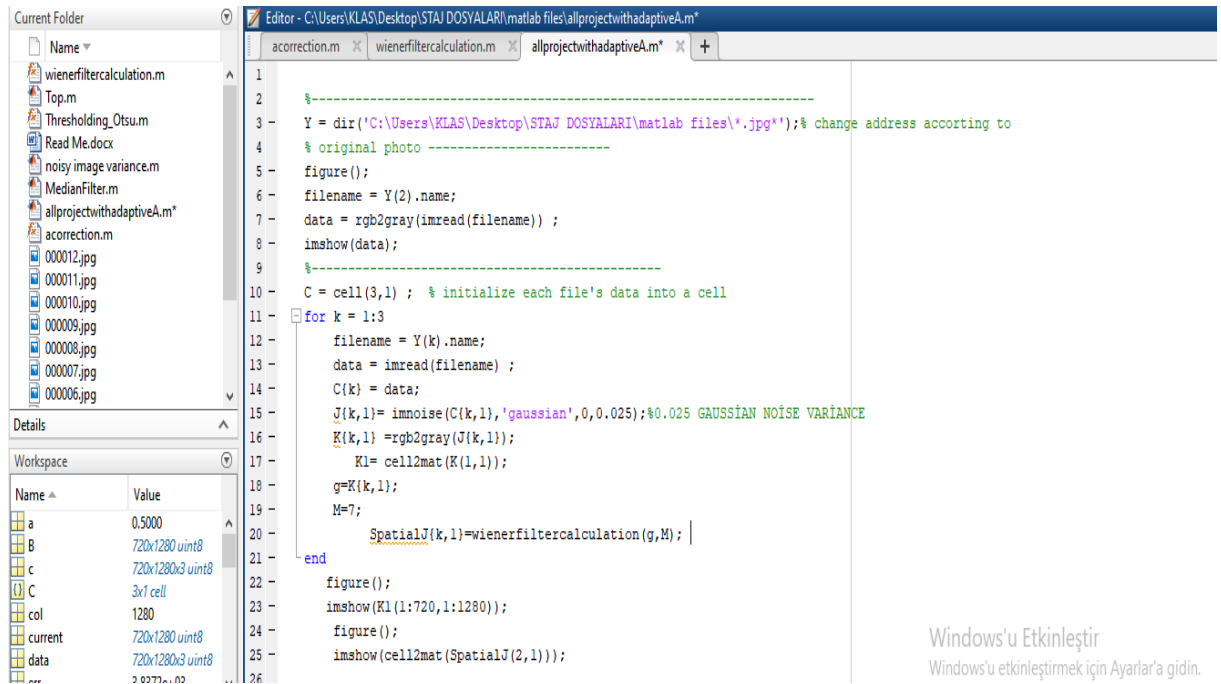


Figure 20: Spatio-Temporal Noise reduction algorithms application on MATLAB part 1

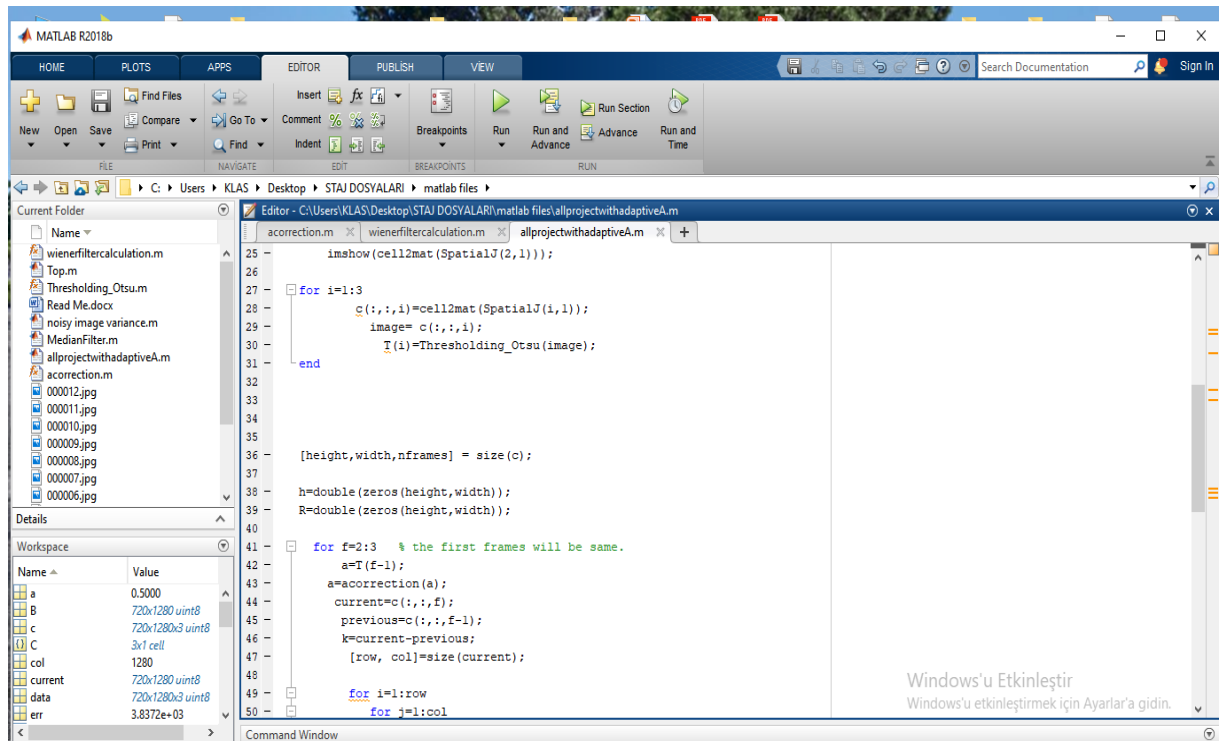


Figure 21: Spatio-Temporal Noise reduction algorithms application on MATLAB part 2

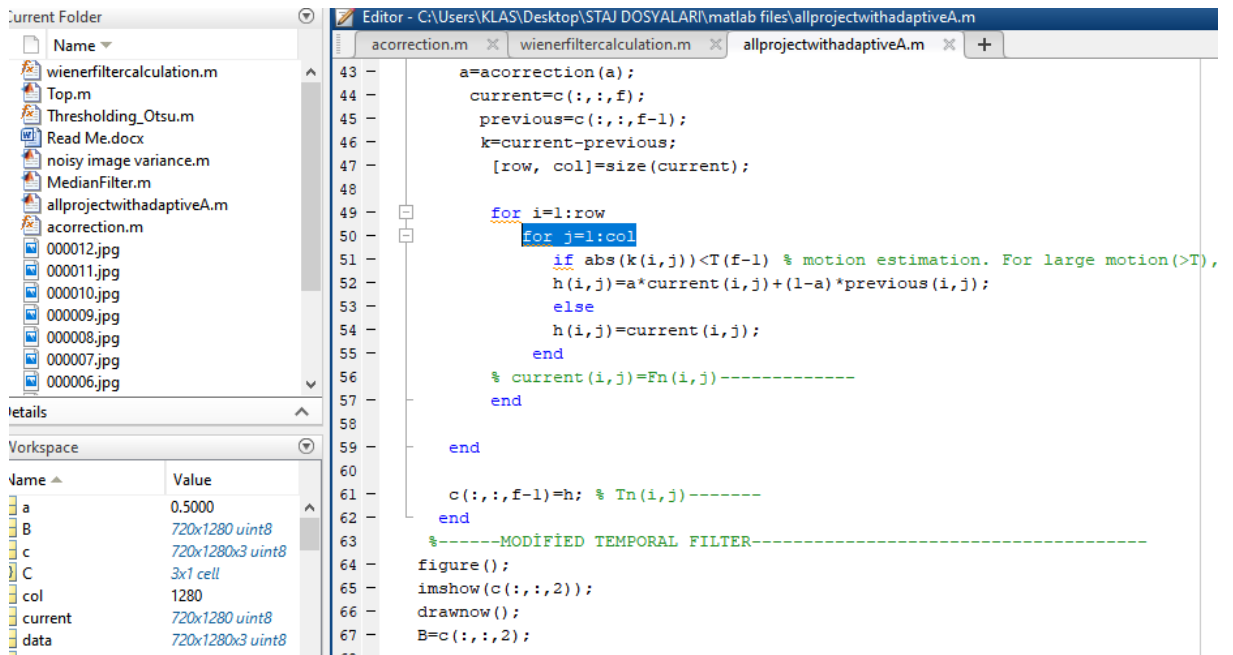


Figure 22: Spatio-Temporal Noise reduction algorithms application on MATLAB part 3

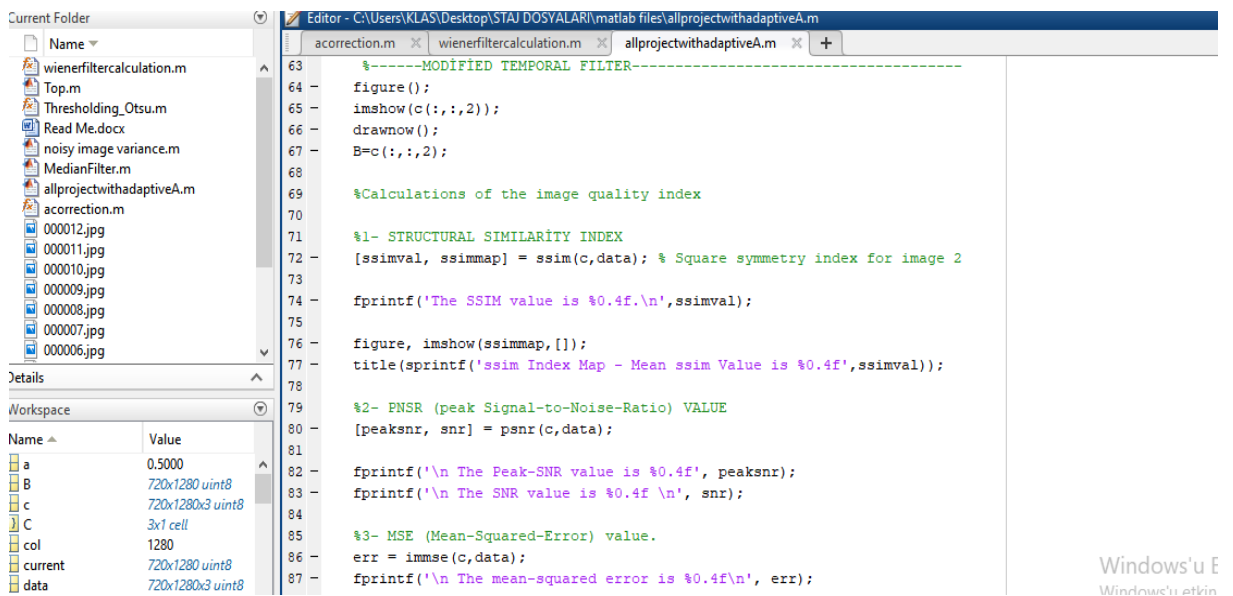


Figure 23: Spatio-Temporal Noise reduction algorithms application on MATLAB part 4

RESULTS

In this part since I need to have at least 3 different frame to implement the MATLAB function, I downloaded a thermal image from the internet and separated the frames from the video. As can be seen from the Figure 24 and Figure 25, there is a noticeable progress we can detect. In Figure 26, The mean of Structural Similarity Index between the grayscale image and the image after Wiener Filter in the First Project is applied. For the same purpose,

In Figure 27, The mean of Structural Similarity Index between the grayscale image and the image after Adaptive Wiener Filter and Recursive Temporal Filter in the Second Project is applied and can be seen there is a noticable difference by SSI results. To be more compact, I showed the results all in one to be detectable easily in Figure 28. Furthermore, The SSI,PSNR and MSE results can be seen in the Figure 29.

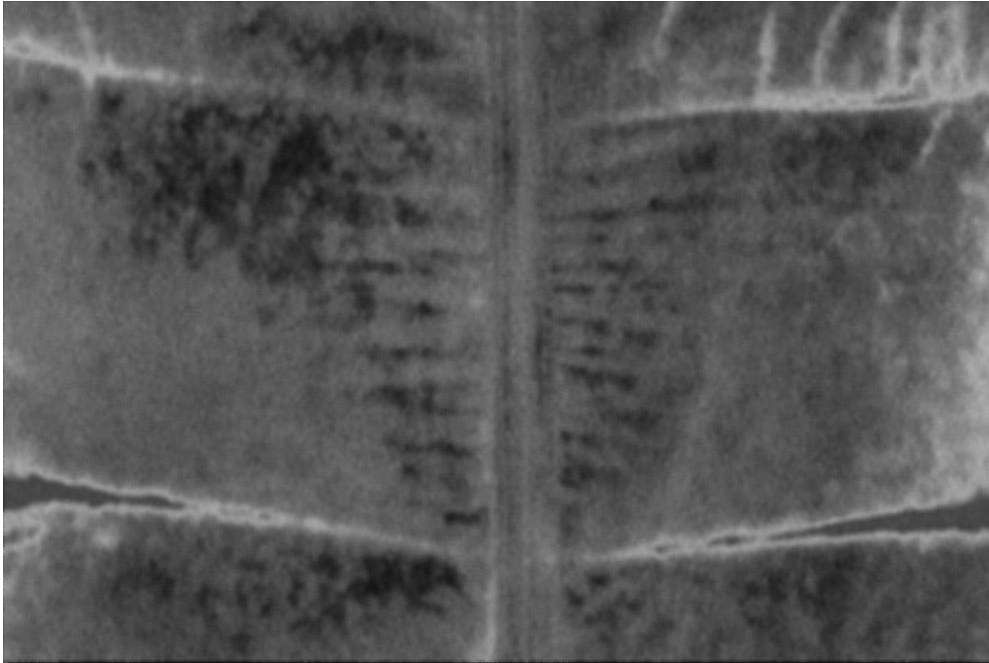


Figure 24: The result after Wiener Filter in First Project is applied.

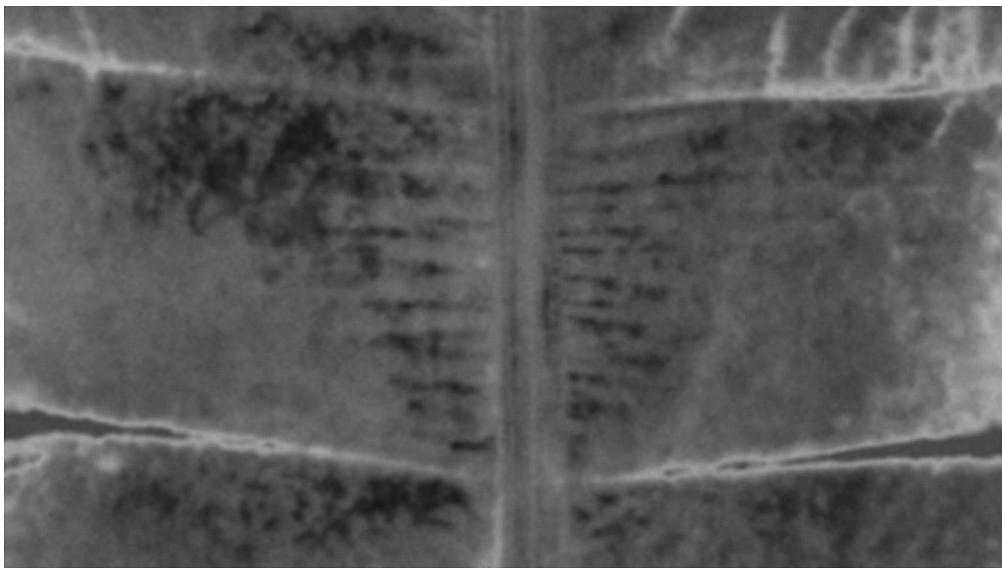


Figure 25: The result after Adaptive Wiener Filter and Recursive Temporal Filter in the Second Project is applied.

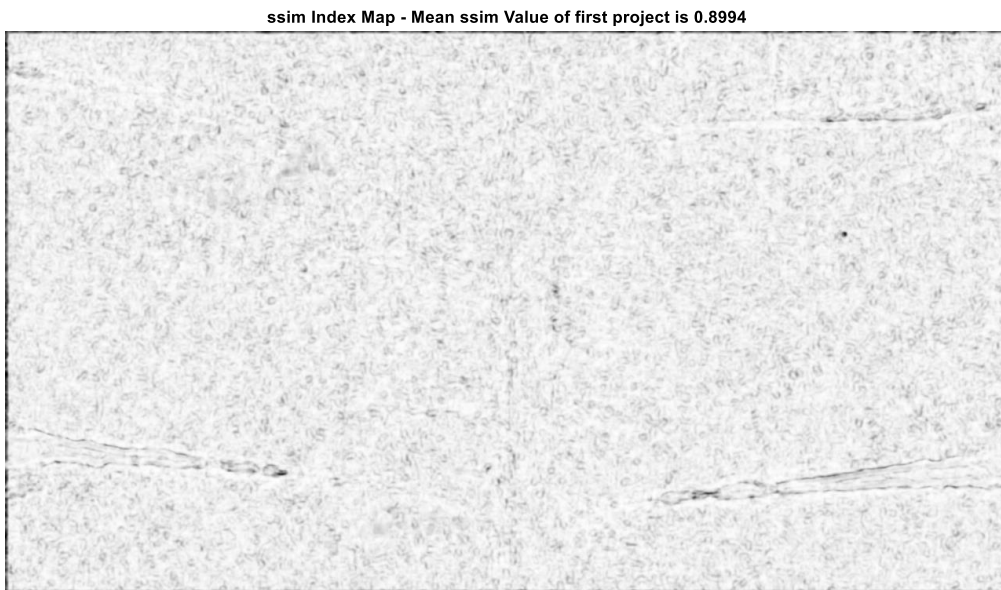


Figure 26: The mean of Structural Similarity Index(SSI) between grayscale image and the result after the Project 1

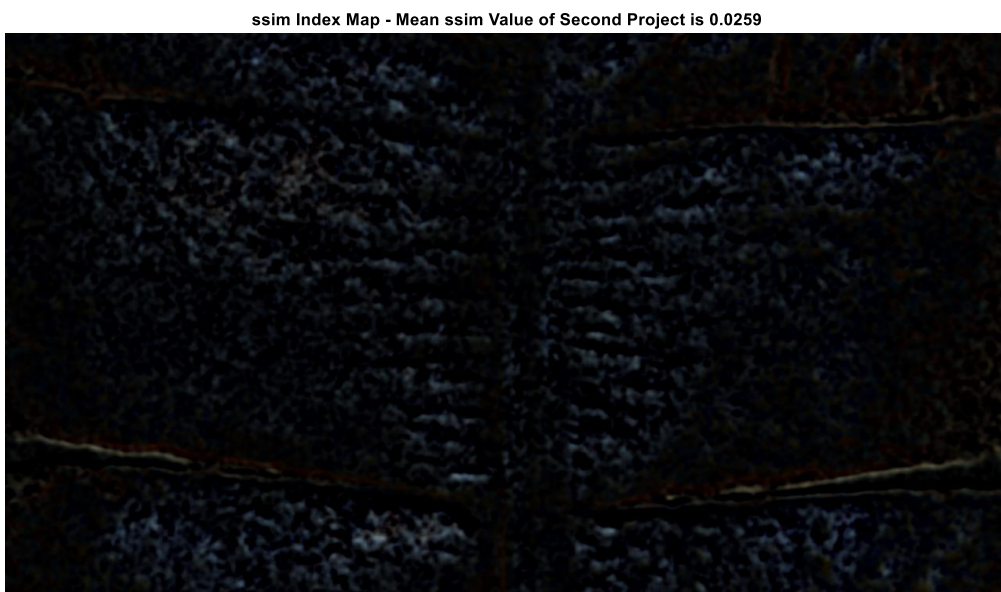


Figure 27: The mean of Structural Similarity Index(SSI) between grayscale image and the result after the Project 2

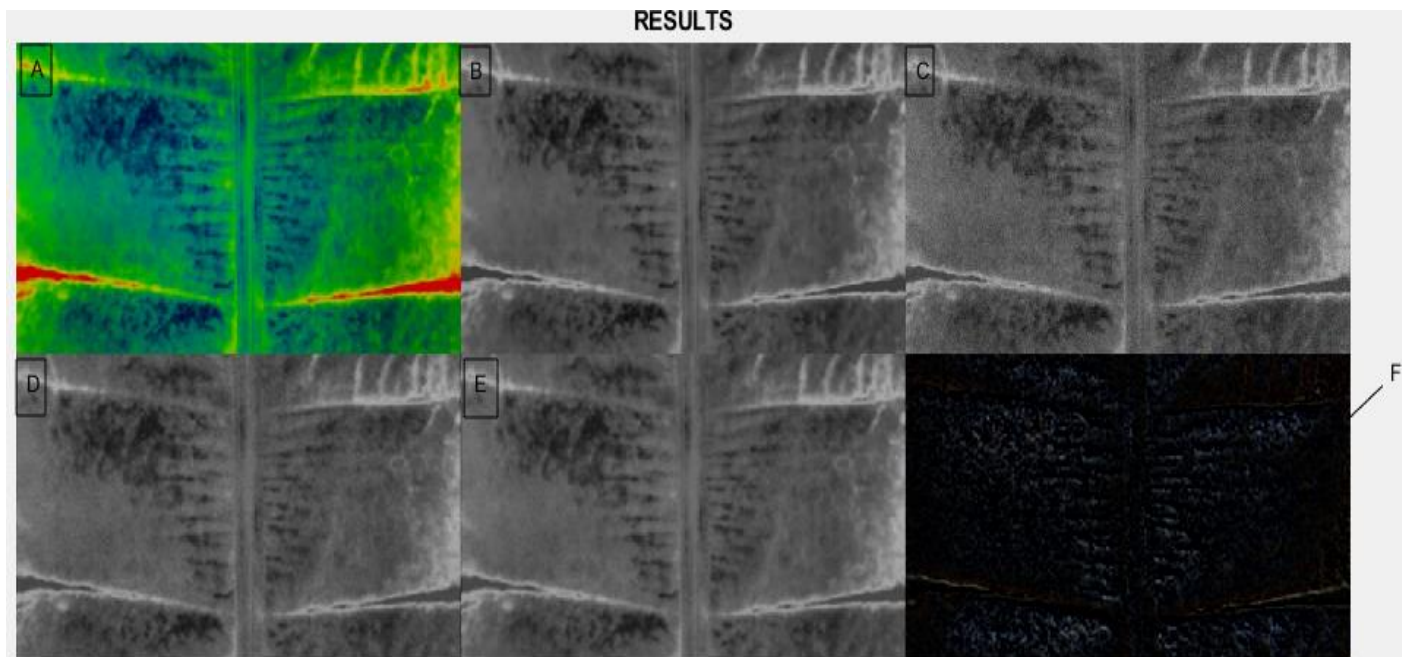


Figure 28: Results of all Project where,

A: Original RGB image, B: Grayscale image, C: Grayscale Image with 0.025 Gaussian additive noise, D: Project 1 Wiener filter result, E: Adaptive Wiener and Adaptive Recursive Temporal Filter, F: Structural Similarity Index results between Figure B and Figure E from Figure27.

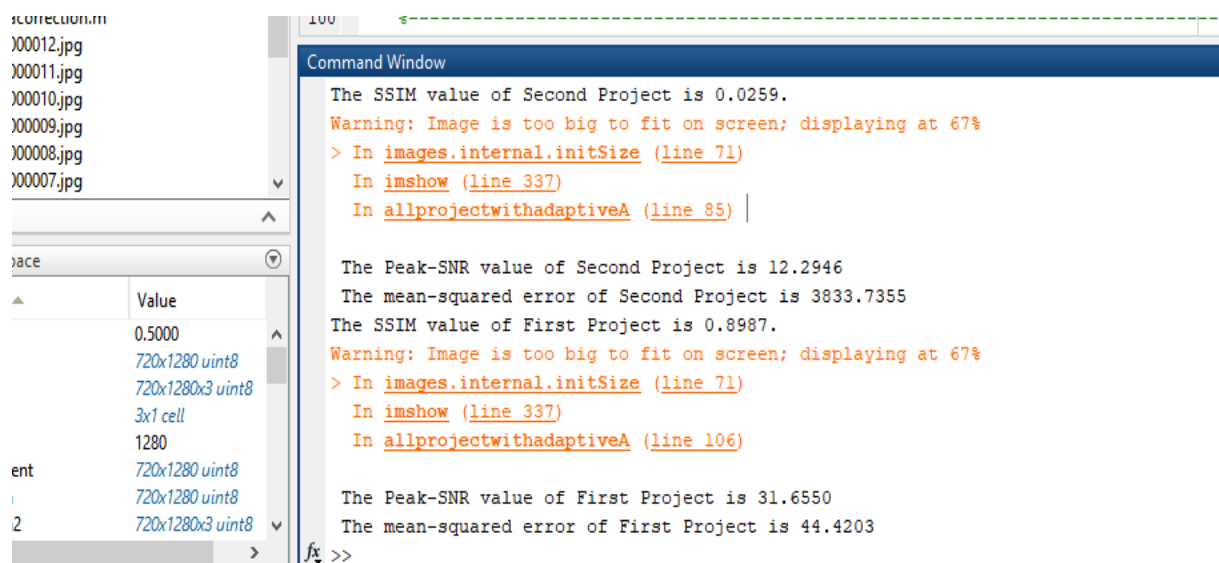


Figure 29: Results of SSI,PSNR and MSE of the First and Second Project

11. COMPARISON BETWEEN MEDIAN FILTER AND SECOND PROJECT

To understand the difference between the traditional methods and the method mentioned in the Second Project, a comparison will be made. The comparison is between Median filter and Spatio-Temporal(Adaptive Wiener Filter and Recursive Adaptive Temporal Filter) Filter.

11.1. COMPARISON FOR 0.025 GAUSSIAN TYPE NOISE

As can be seen from the Figure 30 there is a noticeable difference between Median filter and Spatio-Temporal filter since Median filter is known a good denoise filter for Salt& Pepper noises. The comparison is made for 0.025 Gussian Type noise. Figure 31 also shows the SSIM difference between two filter which demonstrates SSIM valu is 0.1906 for Median Filter and 0.0259 for Spatio-Temporal Filter. In figure 32 and Figure 33, The SSIM, PSNR and MSE performance results between the Median filter and Spatio-Temporal Filter is shown.

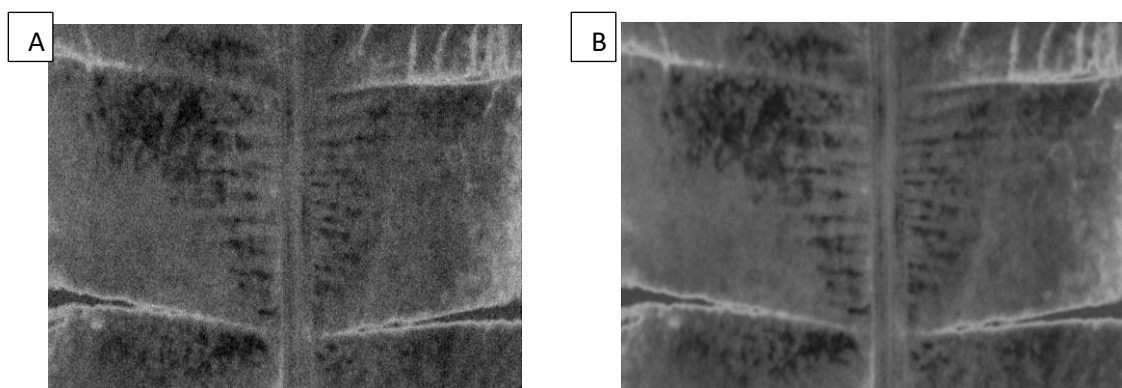


Figure 30: Difference Between Median Filter and the Spatio-Temporal Filter

A: Median Filter results ,

B: Spatio-Temporal Filter Results.

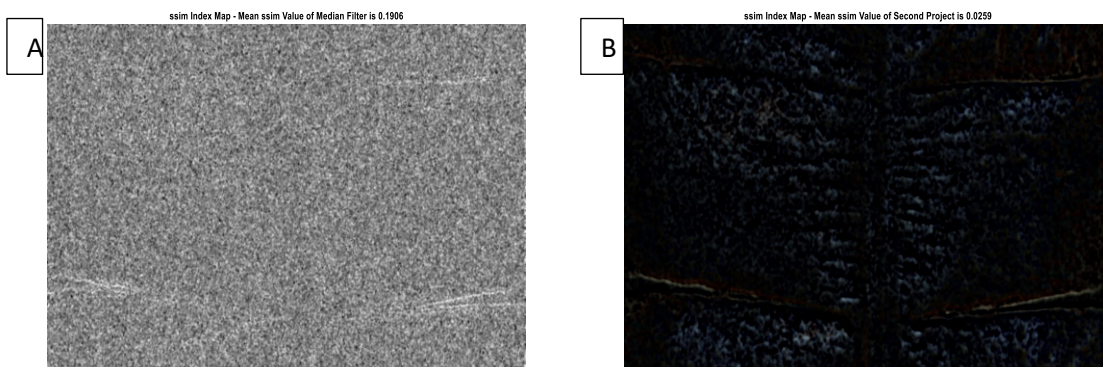


Figure 31:SSIM values for Median Filter and Spatio-Temporal Filter

A: Median Filter SSIM results ,

B Spatio-Temporal SSIM results.

```
The SSIM value of Median Filter is 0.1893.  
Warning: Image is too big to fit on screen; displaying at 67%  
> In images.internal.initSize (line 71)  
   In imshow (line 337)  
   In MedianFilter (line 47)  
  
The Peak-SNR value of Median Filter is 16.5106  
The SNR value of Median Filter is 7.9386  
  
The mean-squared error of Median Filter is 1452.1957  
>>
```

Figure 32: SSIM, PSNR and MSE values of Median Filter.

```
The SSIM value of Second Project is 0.0259.  
Warning: Image is too big to fit on screen; displaying at 67%  
> In images.internal.initSize (line 71)  
   In imshow (line 337)  
   In allprojectwithadaptiveA (line 85) |  
  
The Peak-SNR value of Second Project is 12.2946  
The mean-squared error of Second Project is 3833.7355
```

Figure 33. SSIM, PSNR and MSE values of Spatio-Temporal Filter.

11.2. COMPARISON FOR 0.2 SALT & PEPPER TYPE NOISE

Since median filter is better for Salt & Pepper noise types, the same comparison will be made to prove which filter type's performance is better. It is observed that for the low rate of noise (0.025 Salt and Pepper noise) both Median Filter and Spatio-Temporal Filter is performing well, almost removes all noises from the image as can be seen from the Figure 34. However Since median filter doesn't have much blurring effect on images, the result from the median filter is more clearer than other filter's result. Therefore I decided to increase the noise level 10x(0.2) to observe which of them is better. As can be seen from the In Figure 35, The Median filter is more clearer than the Spatio-Temporal Filter since Spatio-Temporal filter makes images more blurrier. Furthermore, In figure 36 and 37, the SSIM results can be seen to compare in terms of performance. Moreover, In the Figure 38 and 39 the output value of the SSIM, PSNR and MSE can be compared in terms of performance parameters.

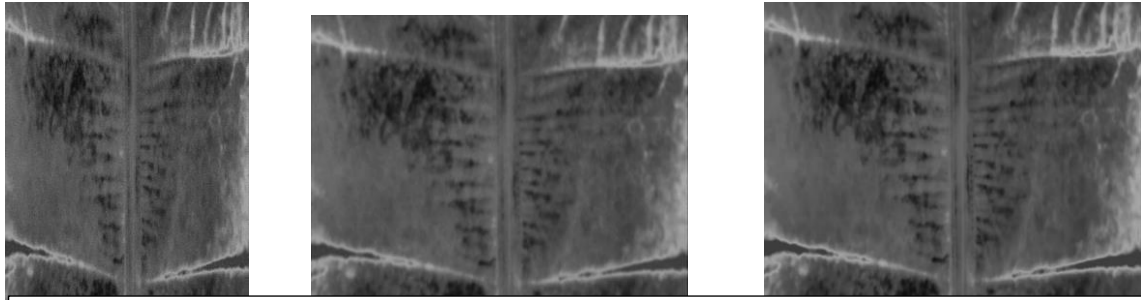


Figure 34: The Salt and Pepper noise results of the Median and Spatio-Temporal filter, respectively.

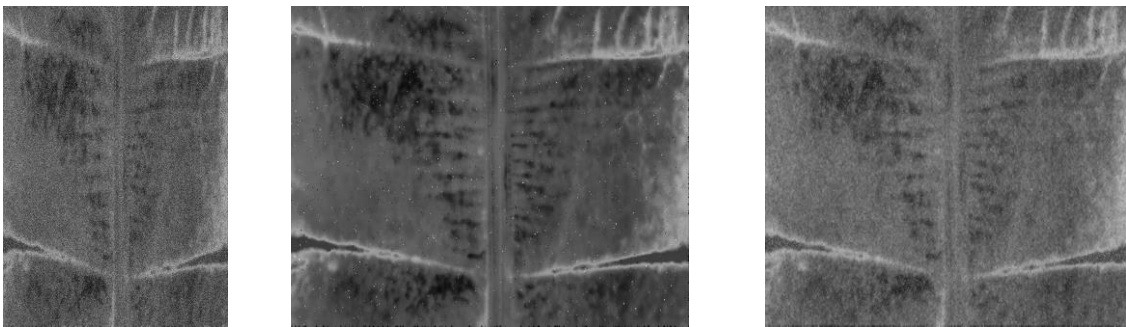


Figure 35: The Salt and Pepper noise results of the Median and Spatio-Temporal filter, respectively.

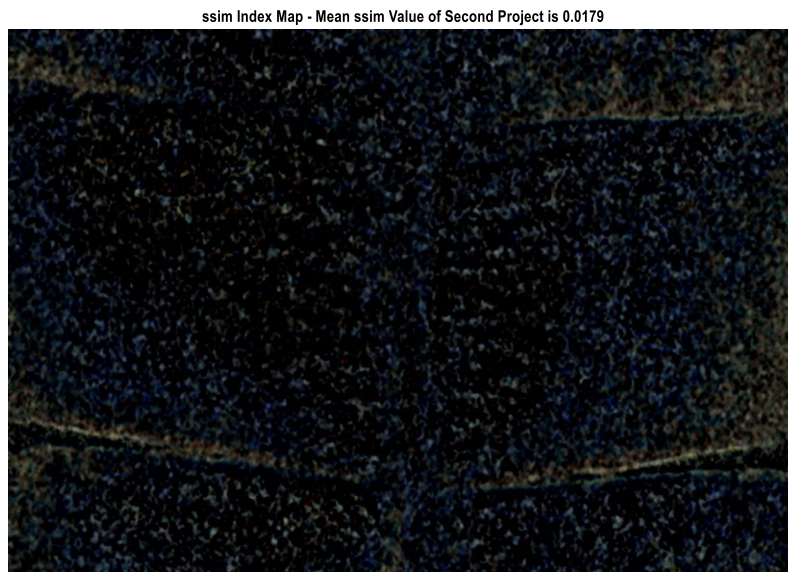


Figure 36: SSIM result (0.0179) of Spatio-Temporal Filter for 0.2 Salt and Pepper Noise

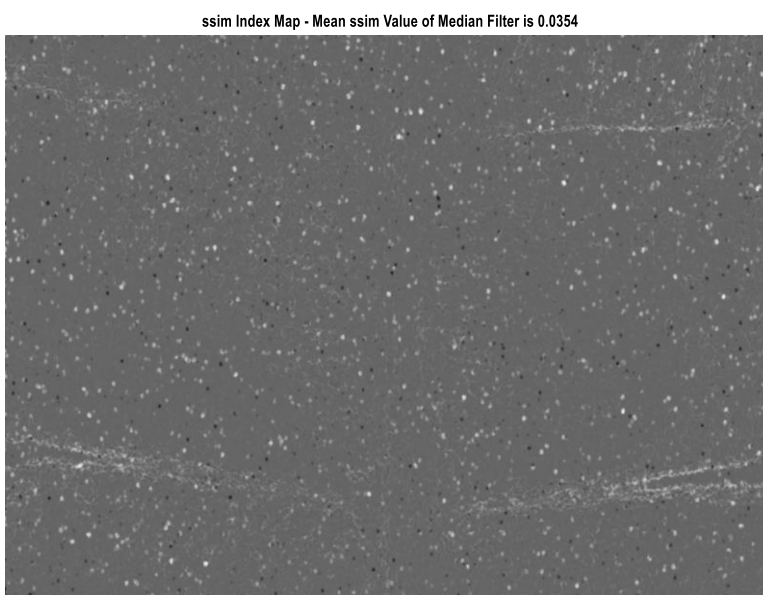


Figure 37: SSIM result (0.0354) for the Median Filter for 0.2 Salt and Pepper Noise

```
The SSIM value of Median Filter is 0.0358.  
Warning: Image is too big to fit on screen; displaying at 67%  
> In images.internal.initSize (line 71)  
   In imshow (line 337)  
   In MedianFilter (line 47)  
  
The Peak-SNR value of Median Filter is 11.4364  
The SNR value of Median Filter is 4.6687  
  
The mean-squared error of Median Filter is 4671.3000  
>> |
```

Figure 38: SSIM, PSNR and MSE results of the Median Filter.

```
The SSIM value of Second Project is 0.0178.  
Warning: Image is too big to fit on screen; displaying at 67%  
> In images.internal.initSize (line 71)  
   In imshow (line 337)  
   In allprojectwithadaptiveA (line 84)  
  
The Peak-SNR value of Second Project is 11.4372  
The mean-squared error of Second Project is 4670.5186
```

Figure 39: SSIM, PSNR and MSE results of the Spatio-Temporal Filter.

12- CONCLUSION

In my summer practice, I have been a part of ASISGUARD for four weeks and learnt how things happen in such a company. Basically, it was a good experience as an engineer and there were a lot of things around for me to observe. I noticed that even if the things that we are taught in school create a basis, there are a lot of practical approaches to learn to be a good engineer.

I have also observed a lot of defence industry products like drone with a gun. I had a chance to see how this type of drone works and examined the mechanism behind it. Also I had the chance to ask how these things work and look at what are the algorithms in software. In addition to this, I learned what kind of hardware is used in this field like FPGAs and MCUs based on ARM STM class.

During my internship, I have learnt using MATLAB which is a great tool for engineering purposes which makes the process fast to see the possible outcomes. I worked on Spatio and Temporal noise reduction on thermal videos with my supervisor engineer. Firstly, I implemented the Wiener filter on noisy thermal images by creating Gaussian noise on MATLAB and denoising it. I understood that the Wiener Filter is good for Gaussian type noises but it makes images blurrier while denoising and it was for only Spatio type noises, so for Temporal noises it wasn't enough. Then, In my second Project I searched for articles so as to make the algorithms

more efficient for both Spatial and Temporal noises. I learnt that By the help of the adaptive Wiener filter and Adaptive Recursive Temporal Filter based on motion compensation methods, we can make images more denoised and less bluer. After implementing this to filter to the noisy images , I used some performance measurement tools like SSIM, PSNR and MSE in order to decide the performance of the system. Lastly, I compared my results with Median filter to see the performance.

All in all, this internship was quite an experience for me. I have observed and learnt a lot about what engineers do in real projects. Also, I observed several fields of electrical and electronics engineering. I became a 4th year student and by the help of this experience I chose the computer option since I liked playing around this kind of things.

13. REFERENCES

[1] Otsu thresholding method. Retrieved from;

<http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html>

[2] Video denoising. Retrieved from;

<https://pdfs.semanticscholar.org/ae02/a2a61a7ca8c210bfc4e409926f227547318b.pdf>

[3] Noise Types on thermal Images;

Image De-noising by Various Filters for Different Noise, Pawan Patidar, Manoj Gupta, Sumit Srivastava, Ashok Kumar Nagawat, International Journal of Computer Applications (0975 – 8887) Volume 9– No.4, November 2010.

[4] E. J. Balster, Y. F. Zheng, and R. L. Ewing, "Feature-based wavelet shrinkage algorithm for image denoising," IEEE Transactions on Image Processing, vol. 14, no. 12, pp. 2024–2039, 2005. [View at Publisher](#) · [View at Google Scholar](#) · [View at Scopus](#)

[5] L. Guo, O. C. Au, M. Ma, and Z. Liang, "Temporal video denoising based on multihypothesis motion compensation," IEEE Transactions on Circuits and Systems for Video Technology, vol. 17, no. 10, pp. 1423–1429, 2007. [View at Publisher](#) · [View at Google Scholar](#) · [View at Scopus](#)

[6] R. Rajagopalan and M. T. Orchard, "Synthesizing processed video by filtering temporal relationships," IEEE Transactions on Image Processing, vol. 11, no. 1, pp. 26–36, 2002. [View at Publisher](#) · [View at Google Scholar](#) · [View at Scopus](#)

[7] E. J. Balster and R. L. Ewing, "Combined spatial and temporal domain wavelet shrinkage algorithm for video denoising," IEEE Transactions on Circuits and Systems for Video Technology, vol. 16, no. 2, pp. 220–230, 2006. [View at Publisher](#) · [View at Google Scholar](#) · [View at Scopus](#)

[8] S.-W. Lee, V. Maik, J.-H. Jang, J. Shin, and J. Paik, "Noise-adaptive spatio-temporal filter for real-time noise removal in low light level images," IEEE Transactions on Consumer Electronics, vol. 51, no. 2, pp. 648–653, 2005. [View at Publisher](#) · [View at Google Scholar](#) · [View at Scopus](#)

[9] L. Yan and Q. Yanfeng, "Novel adaptive temporal filter based on motion compensation for video noise reduction," in Proceedings of the International Symposium on Communications and Information Technologies (ISCIT '06), pp. 1031–1034, Bangkok, Thailand, October 2006. [View at Publisher](#) · [View at Google Scholar](#) · [View at Scopus](#)

[10] Wiener filter: Retrieved From;

<http://www.owl.net.rice.edu/~elec539/Projects99/BACH/proj2/wiener.html>

[11] M. Rakhshanfar and A. Amer, "Motion blur resistant method for temporal video denoising," in Proceedings of the IEEE International Conference on Image Processing (ICIP '14), pp. 2694–2698, Paris, France, October 2014. [View at Publisher](#) · [View at Google Scholar](#)

[12] S. Mishra and P. D. Swami, "Spatio-temporal video denoising by block-based motion detection," International Journal of Engineering Trends and Technology, vol. 4, no. 8, pp. 3371–3382, 2013. [View at Google Scholar](#)

[13] Median Filter: Retrieved From;

<https://www.sciencedirect.com/topics/computer-science/median-filter>

[14] Performance Measurement tools: Retrieved From;

https://www.researchgate.net/publication/27355941_Performance_Evaluation_in_Image_Processing

[15] Noisy image variance: Retrieved From;

<https://ieeexplore.ieee.org/document/788764>

[16] Adaptive Wiener filter: Retrieved from ;

http://www.eas.uccs.edu/~mwickert/ece5655/lecture_notes/ARM/ece5655_chap8.pdf

[17] M. Esche, A. Glantz, A. Krutz, and T. Sikora, "Adaptive temporal trajectory filtering for video compression," IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, no. 5, pp. 659–670, 2012. [View at Publisher](#) · [View at Google Scholar](#) · [View at Scopus](#)