INTRODUCTION TO DIGITAL IMAGE PROCESSING

361.1.4751

EXERCISE 1 - BASIC IMAGE OPERATIONS

Submission Date: 16.11.2022

Noam Atias – 311394357

Chanel Michaeli - 208491787

תוכן עניינים

3 ..... Histogram Manipulation.1

3..... .... Reading the image:

3 .... Histogram Construction:

6..... ....Brightness:

6..... ....Contrast:

8..... ....Quantization:

9 ....Histogram Equalization:

10 ....Histogram Matching:

12 ..... Spatial Filters and Noise .2

12..... ....Read the Image:

12 ....Mean vs Median Filter:

14..... .... Gaussian Filter:

16..... .... Anisotropic Diffusion Filter:

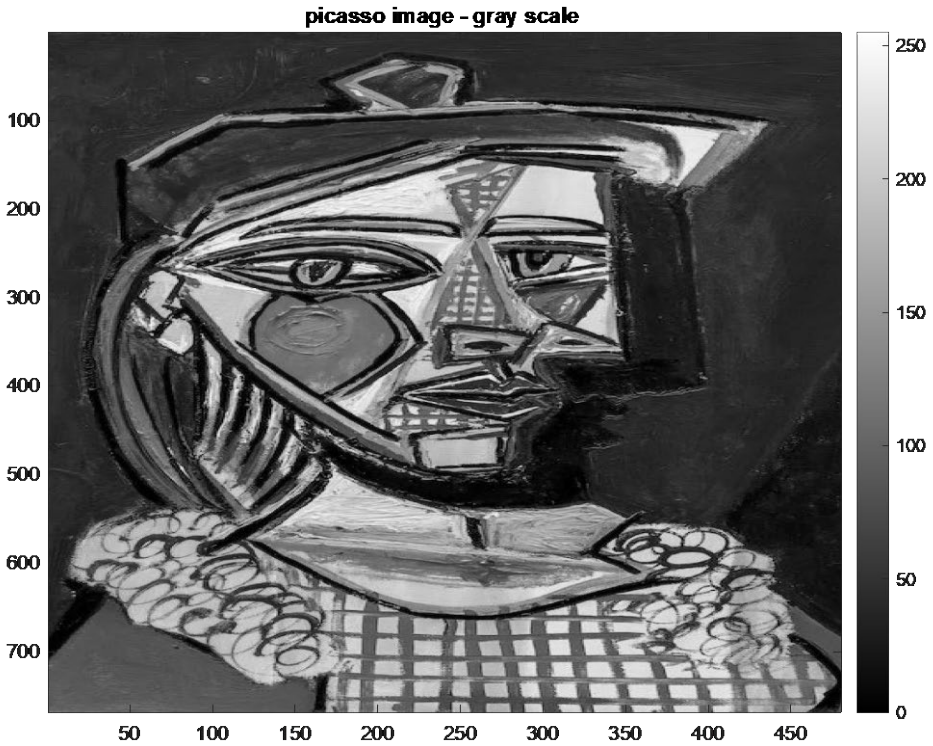
16..... ....Noise Filtering:

Histogram Manipulation.1

Reading the image:

1. ראשית נקרא את התמונה: *'picasso.jpg'* , ולאחר מכן נבצע המרה ל – Gray scale.
נשתמש בפונקציה המובנית *double(rgb2gray()* .

2. נציג את התמונה שקיבלנו על ידי הפונקציה *imagesc()* ונוסיף colormap על ידי שימוש בפונקציה *colormap()*.
התמונה שקיבלנו:



3. נבנה פונקציה משלנו בשם: *dip_GN_imread(file_name)*.
הפונקציה מחזירה את הנרמול של הסקאלה האפורה של התמונה.
הפונקציה מקבלת שם קובץ (תמונה) ומבצעת המרה של התמונה ל – gray scale of type double על ידי שימוש בפונקציה *double(rgb2gray())* , ולאחר מכן מבצעת נרמול כך:

$$\frac{Img - min(img(:))}{max(Img(:)) - min(Img(:))}$$

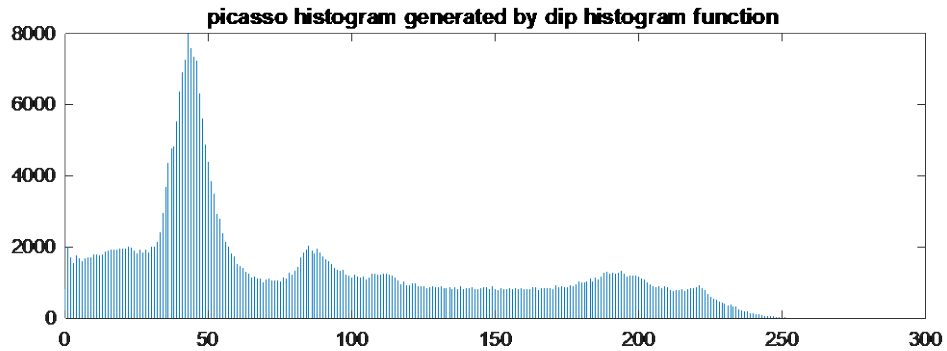
הפונקציה מחזירה את ערכי התמונה בין [0,1]

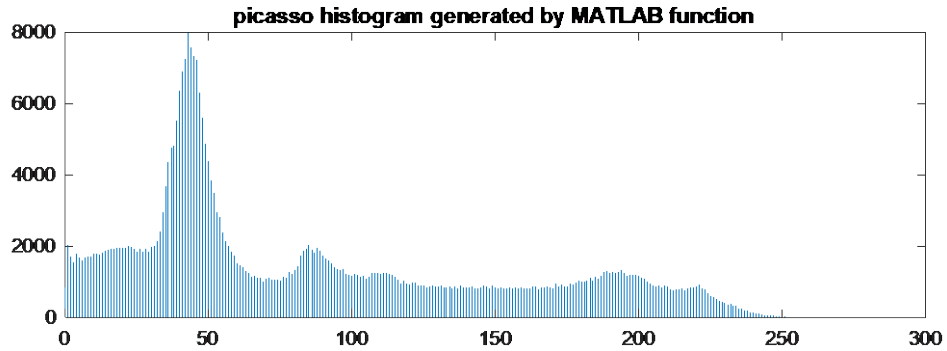
Histogram Construction:

עבור סעיף זה נשתמש בתמונה *'picasso.jpg'* כאשר הוא מסוג *double(rgb2gray())* .

1. נבנה פונקציה בשם: *dip_histogram(img,nbins)*.
הפונקציה מחזירה את ההיסטוגרמה של התמונה, כאשר הפונקציה מקבלת את התמונה ואת מספר ה – bins.

2. נציג את ההיסטוגרמה של התמונה שקראנו בסעיף הקודם כאשר $number\ of\ bins = 256$ ונשווה את התוצאה לפונקציה המובנית של MATLAB: *imhist()*.



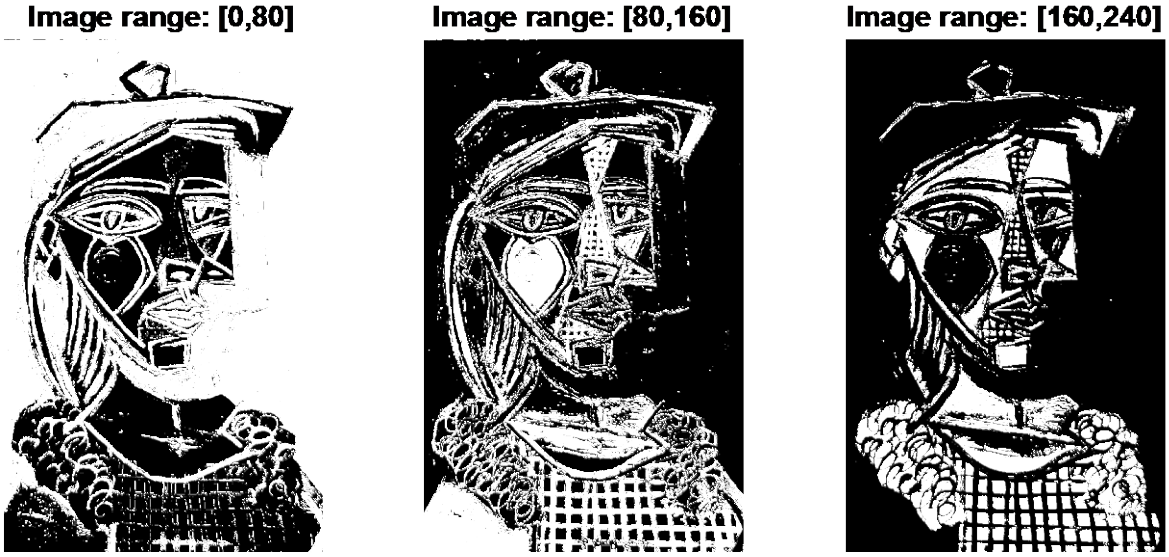


בנוסף נבצע השוואה בין ההיסטוגרמות על ידי שימוש בפונקציה *pdist()* . פונקציה זו מודדת את המרחק האוקלידי בין כמויות הפיקסלים של כל עמודה בהיסטוגרמה אחת לבין ההיסטוגרמה השניה. על מנת לבדוק שהמרחק האוקלידי בין העמודות התאומות הוא 0 נבצע חיבור של האלכסון של המטריצה המחשבת את המרחק האוקלידי.

```
>> D(1:10,1:10)

ans =

         0      1192       891       710       949       865       782       849       877       881
      1192         0       301       482       243       327       410       343       315       311
       891       301         0       181        58        26       109        42        14        10
       710       482       181         0       239       155        72       139       167       171
       949       243        58       239         0        84       167       100        72        68
       865       327        26       155        84         0        83        16        12        16
       782       410       109        72       167        83         0        67        95        99
       849       343        42       139       100        16        67         0        28        32
       877       315        14       167        72        12        95        28         0         4
       881       311        10       171        68        16        99        32         4         0

>> D_trace

D_trace =

     0
```

3. מהסתכלות על ההיסטוגרמה נוכל לומר כי היא מורכבת משלושה גאוסיאנים, כאשר נוכל לומר כי כל גאוסיאן מייצג חלק אחר בתמונה לדוגמא רקע, שיער וכו'.

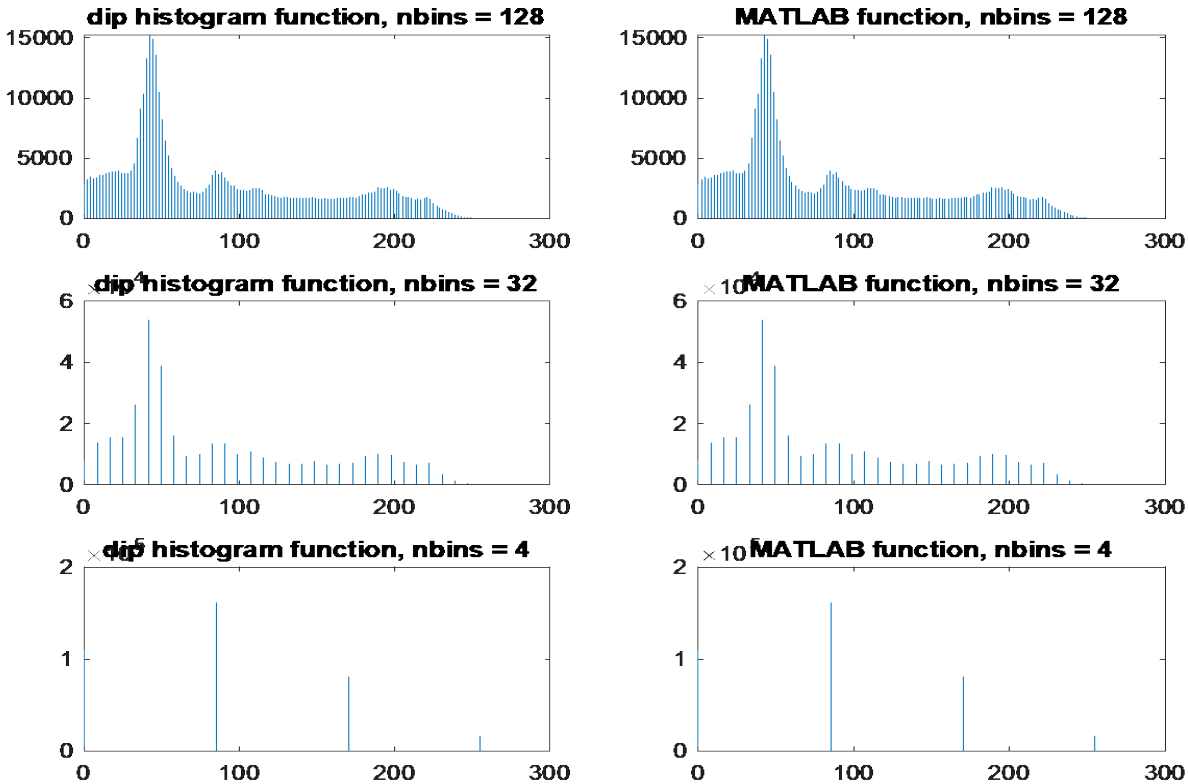
נייצג את התמונה בטווחים השונים לפי ההיסטוגרמה כך שנשמור על ערך 1 של הפיקסל כאשר אנחנו בטווח המסוים ושאר הפיקסלים יקבלו ערך של 0.
נציג את התמונה לפי הטווחים השונים ($[0\,,80]\,,[80\,,160]\,,[160\,,240]$):



נוכל להסיק שעבור התחומים:

- $[0\,,80]$ – הגאוסיאן מייצג ברובו את הרקע.
- $[80\,,160]$ – הגאוסיאן מייצג את הלחיים, האף וחלק מן החולצה והכתף.
- $[160\,,240]$ – הגאוסיאן מייצג את רוב הפנים, את הצוואר ואת המשבצות שבתוך החלוצה.

4. נציג את ההיסטוגרמות עבור $number\ of\ bins = [128, 32, 4]$ ונשווה את התוצאות לפונקציה *imhist()*, המובנית של MATLAB:



נוכל לראות שהתוצאות של ההיסטוגרמות מתוך הפונקציה שבנינו זהות להיסטוגרמות מהפונקציה המובנית של MATLAB.

5. כאשר אנחנו משתמשים במספר גדול יותר של bins , הצורה של ההיסטוגרמה של התמונה ברורה ויחסית נשמרת לדוגמא עבר 128 או 32 עדיין נוכל לראות את הצורה הכללית של ההיסטוגרמה המקורים, כלומר עבור 256, אך עבור מספר קטן כמו 4 קשה לנו לומר איך הצורה הכללית של התמונה נראית, כלומר לא יכולים לדעת כיצד הצורה המקורית של ההיסטוגרמה הייתה בהתחלה.

Brightness:

מעכשיו והלאה נשתמש בגרסה האפורה המנורמלת של התמונה *'picasso.jpg'* על ידי שימוש בפונקציה שבנינו, *dip_GN_imread(file_name)*.

1. נבנה פונקציה לבהירות בשם *adjust_brightness(img,action,parameter)*.
הפונקציה מקבלת תמונה פעולה ופרמטר, כאשר הפעולה הינה חיבור או כפל בגודל הפרמטר.
הפונקציה לוקחת את הערכים של התמונה ובהתאם לפעולה מוסיפה או מכפילה את בפרמטר הנתון.
בנוסף כל ערך של התמונה שלאחר ההכפלה או החיבור בפרמטר הנמצא קטן מ – 0 נגדיר להיות ערך 0 וכל ערך שגדול מ – 1 נגדיר להיות 1 כדי לא לחרוג מהתחום המנורמל.

2. נציג את התמונה המקורית בסקאלה האפורה יחד עם 4 תמונות של התמונה המקורית לאחר הגברה או הורדה של הבהירות:

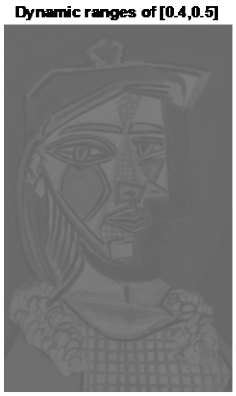
נשים לב לשינוי בבהירות בין התמונה המקורית (השמאלית ביותר) אל מול התמונה המוכפלת בפאקטור של 3 והתמונה הימנית לה שבה הוסף פאקטור של 0.5, כאשר הבהירות בהן גבוהה יותר מהתמונה ואילו עבור התמונה הימנית ביותר ניתן לראות הפחתה בבהירות לאחר הורדה בפאקטור של 0.5 אשר בה יש ירידה בבהירות אל מול התמונה המקורית.

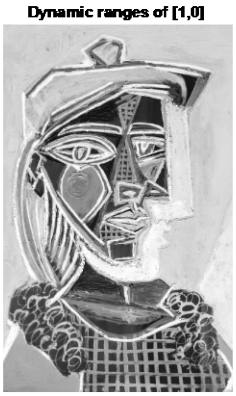
Contrast:

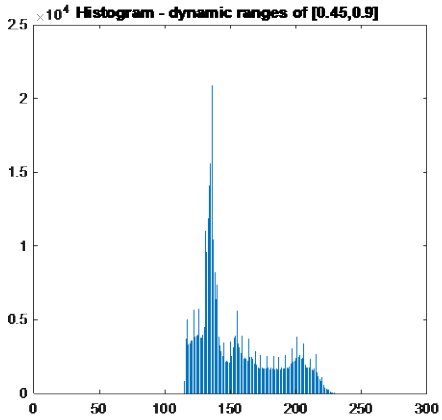
1. נבנה פונקציה לניגודיות התמונה בשם *adjust_contrast(img,range_low,range_high)*.
הפונקציה מקבלת תמונה, טווח גבוה, וטווח נמוך. פרמטרים אלו יקבעו כעת את הטווח הדינמי החדש של התמונה. נשתמש במודל הלינארי כך:

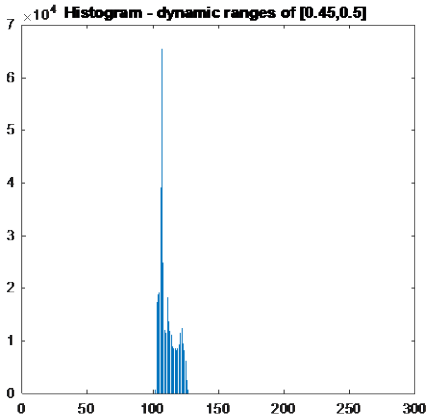
$$contrast_image = img * (range_high - range_low) + range_low$$

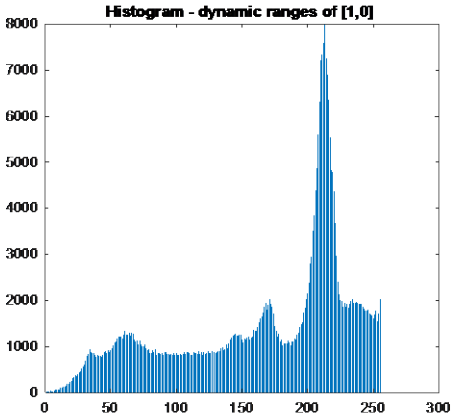
2. נחשב את השינוי של התמונה עבור התחומים: [0 , 1] , [0.4 , 0.5] , [0.45 , 0.9].
נציג את ההיסטוגרמה של כל תמונה ונסביר את ההשפעה של כל תחום בתמונות החדשות.
התמונות לאחר השינוי:











מהתוצאות נוכל לראות שכאשר אנחנו בוחרים תחום של [1 , 0] (התמונה הימנית) אנחנו בעצם הופכים את ערכי התמונה המקורית, כלומר מה שהיה לבן יהפוך שחור וההפך. בנוסף נוכל לראות שההיסטוגרמה היא תמונת מראה של ההיסטוגרמה של התמונה המקורית, וערכי הפיקסלים מתפרסים על כל הטווח שערכי התמונה יכולים לקבל.

עבור הערך של [0.5 , 0.45] (התמונה האמצעית) נקבל תמונה מקווצת יותר, כלומר טווח הערכים של כל פיקסל כעת קטן יותר וקרוב יותר אחד לשני, לדוגמא עבור פיקסל שהיה שחור עם ערך 0 כעת יהפוך להיות אפור יותר שכן הערך הכי נמוך כעת הוא 0.45 ולכן יהיה לנו קשה יותר לראות את התמונה בצורה ברורה יותר כך שחלק מהפרטים נעלמים ברקע עקב הקרבה של הפיקסלים.

עבור הערך של [0.5 , 0.45] (התמונה השמאלית) נקבל תמונה קצת יותר מקווצת מתחום הערכים של התמונה המקורית אך כאן נוכל לראות שפיקסלים שחורים הופכים להיות טיפה פחות שחורים ויותר אפורים ואילו פיקסלים לבנים הופכים להיות קצת יותר אפורים, דבר שמדגיש לנו פרטים שנבלעו אחד בשני ונוכל לראות את התמונה קצת יותר ברורה.

3. נציג את השינוי בתמונות לאחר שנבצע שינוי בצורה לא לינארית עבור התחומים $[0.2\ ,0.8]\ ,[0.45\ ,0.9]$:

None linear mapping of range [0.2,0.8]

None linear mapping of range [0.45,0.9]

מכיוון שכעת אנחנו ממפים כל פיקסל בצורה לא לינארית שנקבל שיש לנו יותר פיקסלים בצבעים אחידים מכיוון שכעת נתנו לכל פיקסל את ערך הקצה בקרוב ביותר אליו בטווח הרצוי ובמקרה בו הם חרגו מן הטווח הרצוי.

Quantization:

בסעיף זה נשנה את ערכי הפיקסלים של התמונה לפי רמות שונות של קוונטיזציה, כלומר מספר רמות האפור המתקבלות עבור $[6\ bit, 4\ bit, 2\ bit, 1\ bit] = [64\ ,16\ ,4\ ,1]$.
נציג את התמונות שקיבלנו:

Picasso after quantization of 6 bits

Picasso after quantization of 4 bits

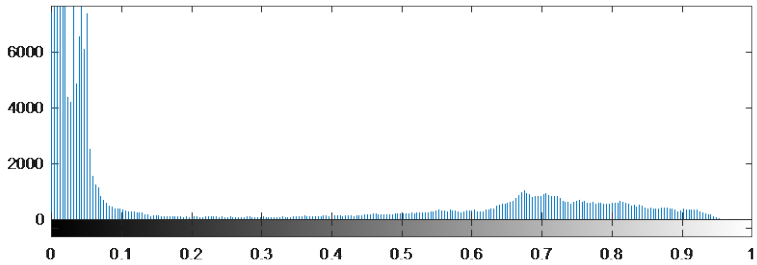
Picasso after quantization of 2 bits

Picasso after quantization of 1 bits

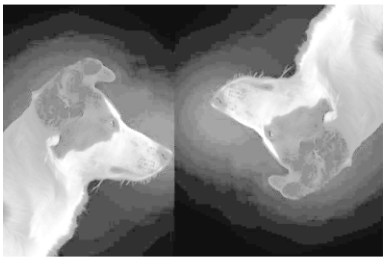
נשים לב, שככל שמספר הביטים של ייצוג התמונה גדול יותר, כלומר רמות האפור שכל פיקסל יכול להימצא בהן גדול יותר נקבל תמונה שמורכבת מהמון גוונים של אפור ומאפשר לנו לראות את התמונה בצורה חלקה יותר ופחות בגוונים של שחור ולבן בלבד.
לעומת זאת אם נסתכל על ייצוד של 2 ביטים בלבד נקבל שכל פיקסל קיבל ערך של 0 או 255 כתלות בקרבה לערך של 0 או 255, כלומר עבור פיקסלים עם ערכים מ – 0 עד 128 הפיקסל יהפוך שחור יותר כלומר עם ערך 0, ועבור פיקסלים עם ערכים מ – 129 עד 255 הפיקסל יהפוך ללבן יותר, כלומר עם ערך 255.

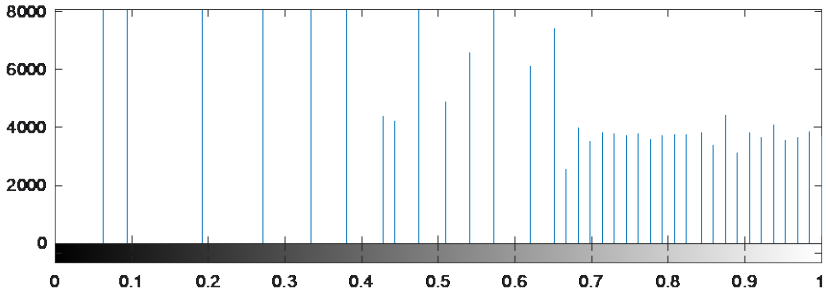
Histogram Equalization:

1. נקרא את התמונה *'dog.jpg'* נהפוך אותו ל – *type double* וננרמל אותה על ידי הפונקציה *dip_GN_imread(file_name)*.
2. נשתמש בפונקציה הבנויה של MATLAB, *histeq()* אשר מאזנת את ההיסטוגרמה של התמונה.
3. כעת נציג התמוה ואת ההיסטוגרמה שלה:



נציג את התמונה החדשה לאחר הפעלת הפונקציה המובנית של MATLAB.

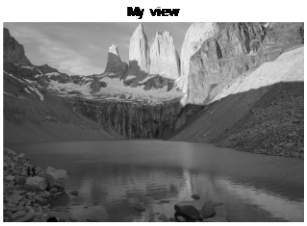


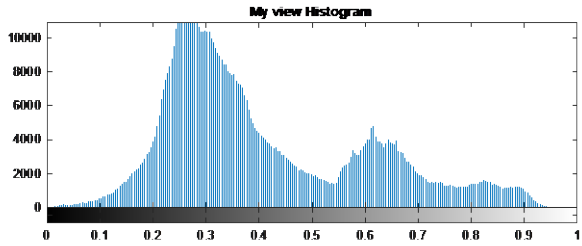


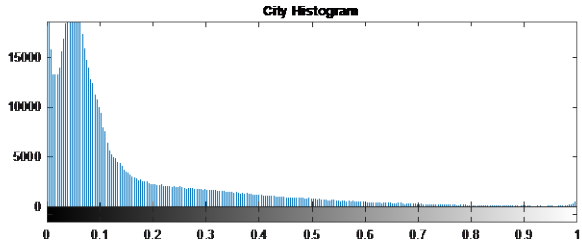
4. נשים לב כי השימוש מהשימוש בפונקציה לא קיבלנו תוצאות טובות מכיוון שהרקע כעת פחות ברור אך עם זאת נוכל לראות את האוזניים של הכלב בקלות יותר. השטחת ההיסטוגרמה גרמה לכך שנוכל להבחין בהבדלים קטנים ברקע מה שפגע ברצף האחידות של הרקע.
הפתרון המוצע לבעיה זו היא בכך שנעלה על פיקסל בחזקה של 0.75 וכך הרקע יהיה אחיד ועדיין יאפשר לנו לזהות את אוזני הכלב. נציג את התמונה לאחר מימוש הפתרון:

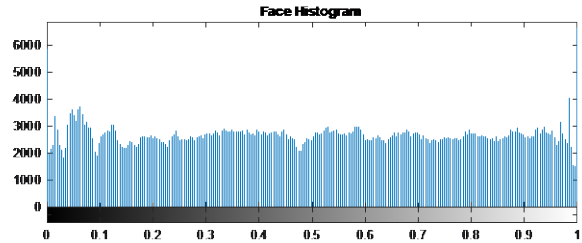
Histogram Matching:

1. ניקח תמונה שלנו מהמחשב, נבצע לה המרה ל – *gray scale type double* וננרמל אותה לערכים של [0 , 1] על ידי *dip_GN_imread(file_name)*.
2. נקרא את התמונה *'city.jpg'* נבצע לה המרה ל – *gray scale type double* וננרמל אותה לערכים של [0 , 1] על ידי *dip_GN_imread(file_name)*.
3. נקרא את התמונה *'face.jpg'* נבצע לה המרה ל – *gray scale type double* וננרמל אותה לערכים של [0 , 1].
4. נציג את התמונות ואת ההיסטוגרמות שלהן:

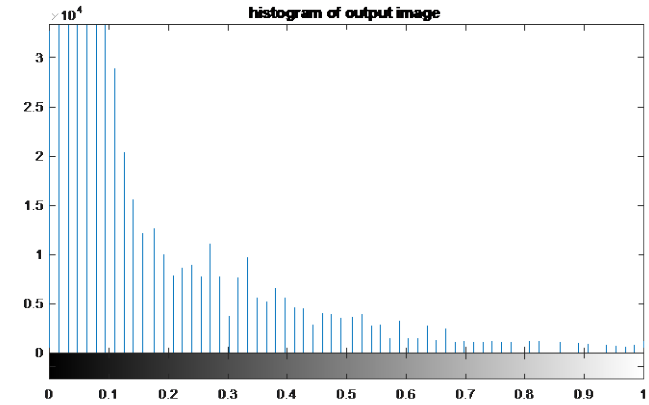


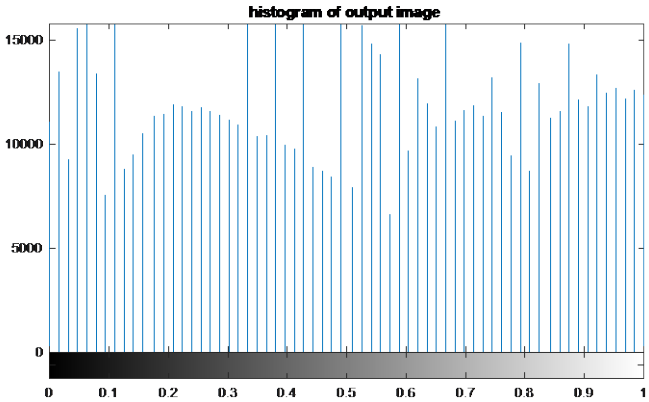






5. נשתמש בפונקציה *imhistmatch()*.
6. נציג את התמונות החדשות וההיסטוגרמות לאחר שימוש בפונקציה המובנית:





7. נוכל לראות שההיסטוגרמות שקיבלנו דומות להיסטוגרמות של התמונות אליהן השוונו את התמונה שלנו. עבור ההשוואה של התמונה שלנו יחד עם התמונה של העיר נקבל כי ההיסטוגרמה עבור ערכים נמוכים יותר ופחות ערכים גבוהים, כלומר תמונה כהה יותר, מכיוון שהתמונה של העיר היא בגוונים כהים יותר מאשר בהירים.
לעומת זאת עבור השוואה עם תמונת הפנים נוכל לראות שהתמונה קיבלה היסטוגרמה מרוחה יותר בדומה להיסטוגרמה של תמונת הפנים, כלומר נוכל לראות פרטים נוספים שהיו כהים יחסית ונבלעו ברקע.

2. Spatial Filters and Noise

Read the Image:

בסעיף זה קראנו את התמונה dog.jpg :

נרמלנו אותה על ידי שימוש בפונקציה שבנינו בשאלה הקודמת.
בתמונה זאת נשתמש בסעיפים הבאים.

Mean vs Median Filter:

1. <u>פילטר ממוצע :</u> על מנת לקבל את פילטר הממוצע בנינו פונקציה המקבלת כקלט את התמונה ואת גודל הפילטר - *mean_filter(img,k)* .
פונקציה זו מייצרת פילטר בגודל k x k שמטרתה להחליף כל פיקסל בתמונה בממוצע של השכנים שלה.
על מנת להתמודד עם גבולות התמונה, ריפדנו באפסים את התמונה כפונקציה של גודל הפילטר. בנוסף, כדי לוודא שאנו מחלקים את הסכום של כל השכנים במספר הפיקסלים לא כולל אלו שנועדו לריפוד, יצרנו פילטר נוסף של אחדות ואפסים כאשר 0 משמעו פיקסל שנועד לריפוד ו- 1 משמעו פיקסל השייך לתמונה המקורית. באופן כזה, הצלחנו לבצע למצע כל פיקסל על ידי השכנים שלו בלבד, גם בגבולות התמונה.
הפלט של הפונקציה הוא בגודל התמונה המקורית.

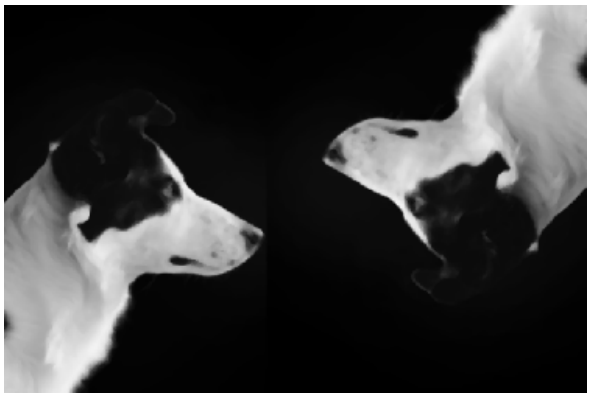
2. <u>פילטר חציון :</u> פילטר זה בנוי על ידי פונקציה שהקלט שלה הוא תמונה וגודל הפילטר - *median_filter(img,k)* .
פילטר זה בנוי בצורה דומה לפילטר הממוצע , גם פה ריפדנו באפסים על מנת להתמודד עם גבולות התמונה. על מנת לבצע את החציון, הפכנו את ה – kernel לווקטור (כאשר בגבולות התמונה נזהרנו לא לכלול את האפסים שנועדו לריפוד, על ידי פילטר העזר שבנינו בדומה למקודם), ביצענו מיון לווקטור על מנת לגשת לחציון. הפלט של הפונקציה הוא בגודל התמונה המקורית.

3. נציג את התמונה לאחר שעברה פילטר ממוצע ופילטר חציון בגדלים שונים:

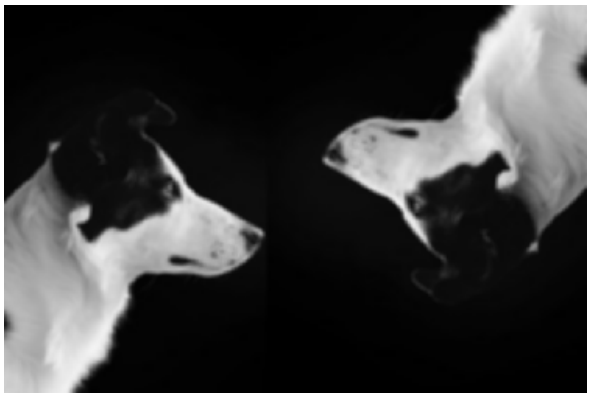
Dog image filtering - median filter 3x3

Dog image filtering - mean filter 3x3

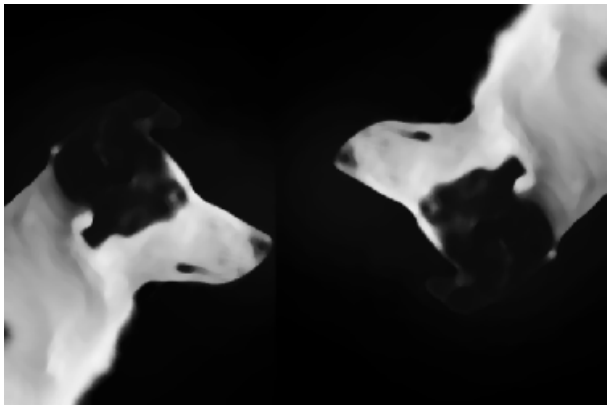
Dog image filtering - median filter 5x5



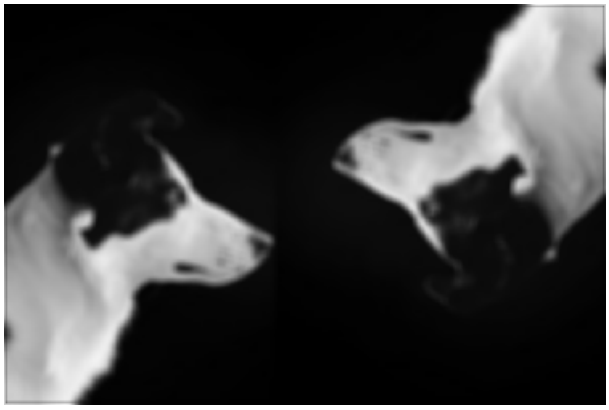
Dog image filtering - mean filter 5x5



Dog image filtering - median filter 9x9



Dog image filtering - mean filter 9x9



נבחין כי ככל שגודל הפילטר גדל, עבור שני סוגי הפילטרים, התמונה נעשית יותר מטושטשת ופחות ברורה. דבר זה מתיישב עם ההיגיון כי ככל שגודל הפילטר גדל כך המיצוע שנעשה על כל פיקסל תלוי במספר גדול יותר מן השכנים שלו. עבור פילטר בגודל 3x3 המיצוע נעשה רק על ידי השכנים הקרובים ביותר אך ככל שגודל הפילטר גדל, כך מתחשבים בשכנים רחוקים יותר המחזיקים מידע פחות רלוונטי לעומת השכנים הצמודים ביותר.
הבחנה נוספת היא ההבדל בתמונה שעברה פילטר חציון לעומת התמונה שעברה פילטר ממוצע. נשים לב כי התמונות לאחר פילטר חציון ברורות וחדות יותר לעומת התמונות לאחר פילטר ממוצע, שנראות מטושטשות יותר. הבדל זה ניכר לעין בעיקר בתמונות בהם גודל הפילטר גדול יותר.
דבר זה נובע מכך שפילטר ממוצע הוא מעין מסנן מעביר נמוכים (LPF) , ולכן הוא מטשטש את השפות ונוצרת תמונה חלקה יותר.

Gaussian Filter:

1. בנינו פונקציה המייצרת פילטר גאוסיאן המקבלת כקלט את התמונה, גודל הפילטר k וsigma .
נתון כי מטריצת הקוואריאנס היא $\begin{pmatrix} sigma & 0 \\ 0 & sigma \end{pmatrix}$ ולכן יכולנו להסיק כי השונות המשותפת של המשתנים היא אפס, כך שניתן לחשב את הגאוסיאן של הפילטר באופן הבא:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

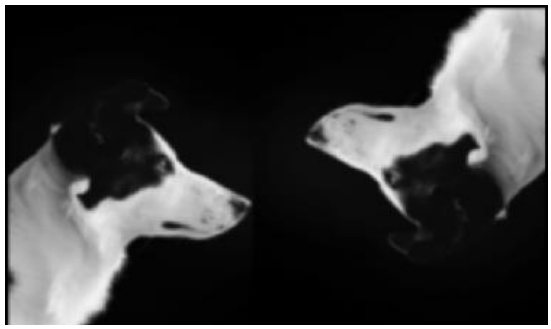
כאשר σ הוא קלט של הפונקציה ו-(x,y) הם הקואורדינטות של הפילטר.
בנוסף, נרמלנו את הפילטר כך שהסכום יהיה 1 וריפדנו באפסים כדי להתמודד עם גבולות התמונה.

2. נציג את התמונה לאחר פילטר גאוסיאן בגדלים שונים עם σ שונה:

Gaussian Filter k=3 sigma=1.7

Gaussian Filter k=3 sigma=0.2

Gaussian Filter k=9 sigma=1.7

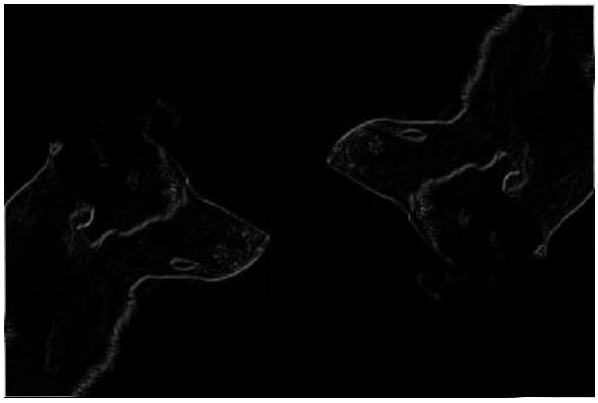


Gaussian Filter k=9 sigma=0.2

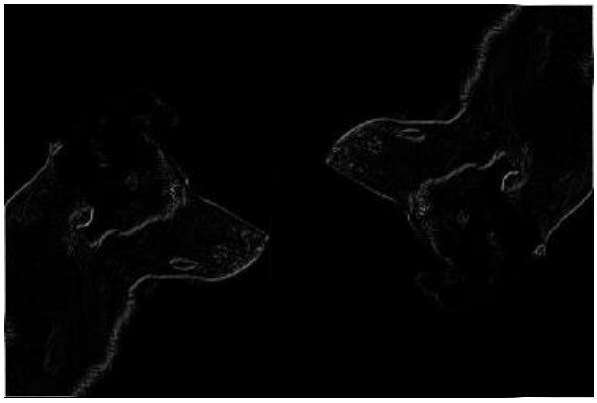
נבחין כי ככל שמספר הפיקסלים בפילטר גדל כך נראה תמונה מרוחה יותר, הניגודיות במעברים בין הצבעים קטנה ולכן התמונה פחות חדה.
זאת מאחר שפילטר זה מבצע מיצוע בין הפיקסל לשכניו כאשר ישנו משקול לכל פיקסל – הפיקסל המרכזי מקבל משקל מקסימלי, שכניו הצמודים לו מקבלים משק קטן יותר ושכנים רחוקים יותר מקבלים משקל קטן עוד יותר, והשונות קובעת באיזו מידה מתחשבים בשכנים מרוחקים.
לכן, עבור σ גדולה יותר אנו מקבלים תמונה יותר מטושטשת כי השכנים הרחוקים יותר מקבלים משקל גדול יותר.

3. נבצע חיסור של התמונה המקורית מהתמונה שעברה סינון גאוסי בערך מוחלט ונציג את התמונה החדשה :

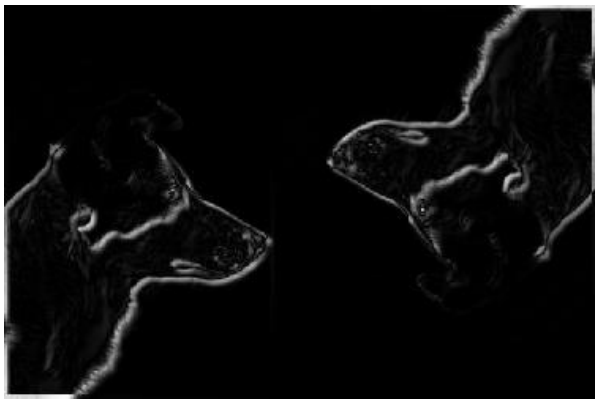
Subtracted image filtered k=3 sigma=1.7



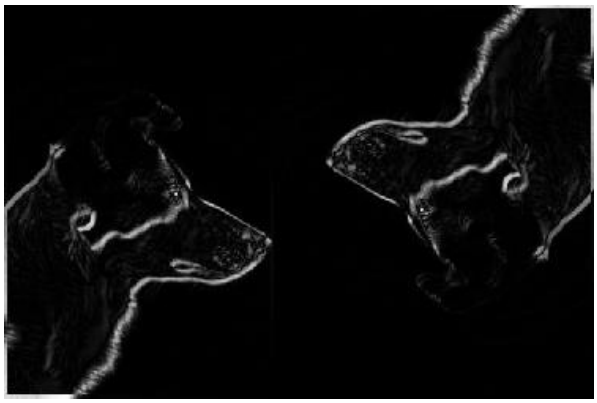
Subtracted image filtered k=3 sigma=0.2



Subtracted image filtered k=9 sigma=1.7



Subtracted image filtered k=9 sigma=0.2



נשים לב כי ככל שגודל הפילטר גדל כך השפות של התמונה יותר בולטות, דבר המתיישב עם ההיגיון מאחר שככל שכמות הפיקסלים בפילטר גדולה כך השפות של התמונה נעשות פחות חדות ולכן בחיסור של התמונה המסוננת מהתמונה המקורית רואים שפות ברורות יותר, שהם התדרים הגבוהים של התמונה המקורית.

Anisotropic Diffusion Filter:

נשתמש בפונקציה המובנת של מטלב *imdiffiusefilt( )* על מנת לבצע את הפילטר על התמונה ונציג את התמונה המתקבלת:

Anisotropic difiusion filter on dog image

מטרת הפילטר היא סינון רעשים מבלי לפגוע בחלקים המשמעותיים של התמונה. נשים לב כי מטרה זו הושגה בתמונה מאחר לעומת התמונה המקורית, תמונה זו נראית חלקה יותר, אך נראה כי החלקים המשמעותיים בתמונה שהם הכלבים לא נפגעו כתוצאה מהמסנן.

Noise Filtering:

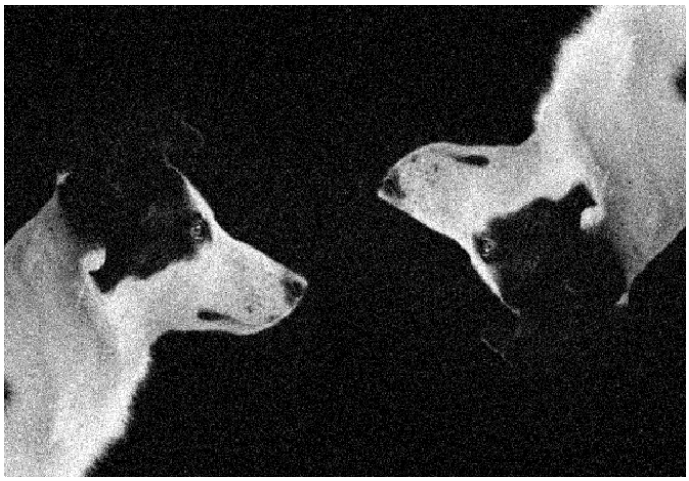
1. תחילה יצרנו שלוש תמונות חדשות על ידי הוספה של שלושה סוגי רעש לתמונה המקורית: *'salt & pepper', 'gaussian' , 'speckle'*.

2. מימשנו את הפילטרים שבנינו (פילטר ממוצע, פילטר חציון, פילטר גאוסי ופילטר Anisotropic Diffusion) על כל אחת מהתמונות, עבור גדלי kernel k=3,k=9 .

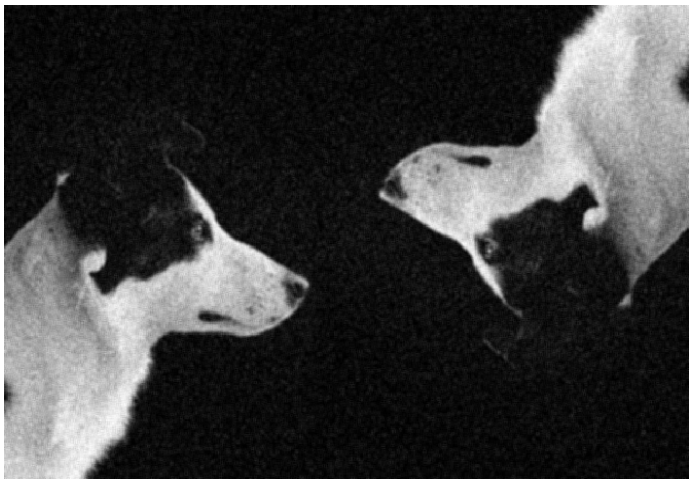
3. נציג את התמונות המורעשות והתמונות לאחר סינון:

תמונות עם רעש גאוסי לפני ואחרי סינון (גודל מסנן k=3):

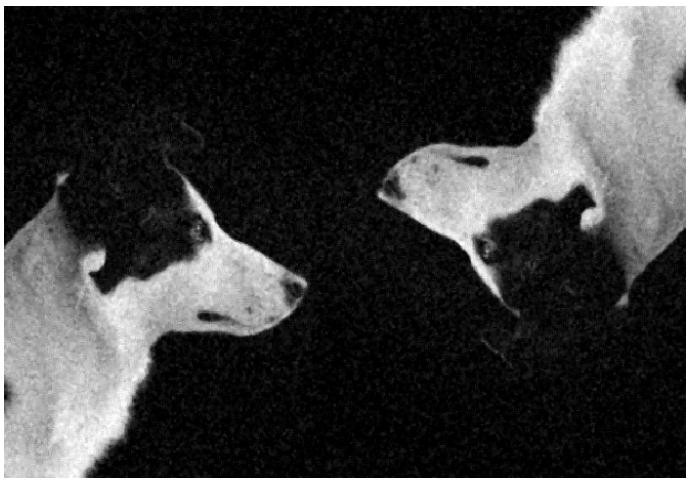
Gaussian noised image



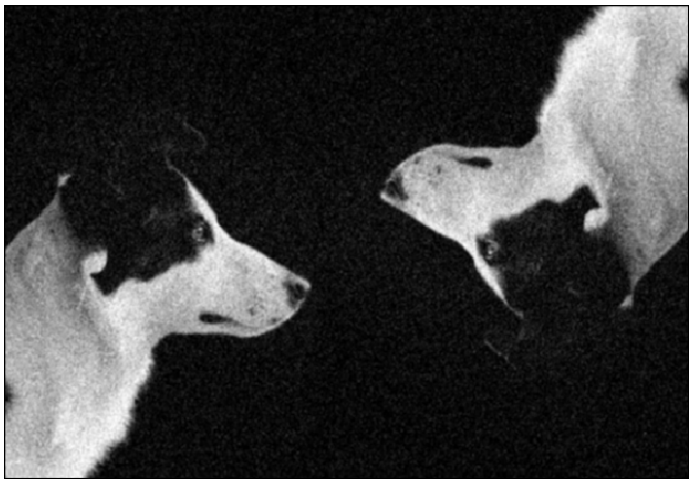
Gaussian noise- mean filter k=3



Gaussian noise- median filter k=3

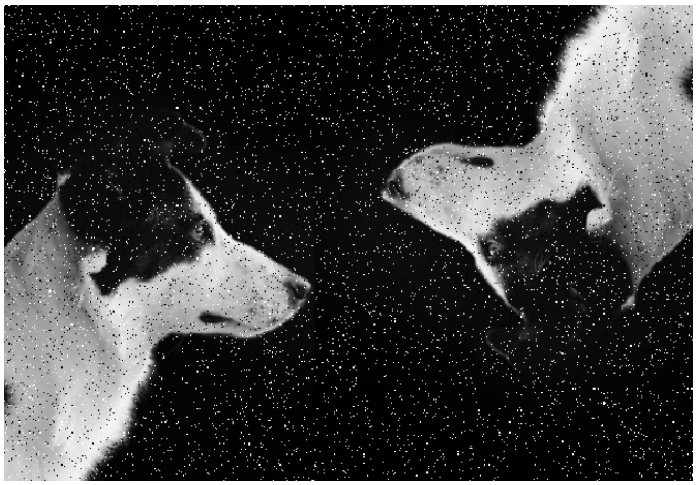


Gaussian noise- gaussian filter k=3,sigma=1

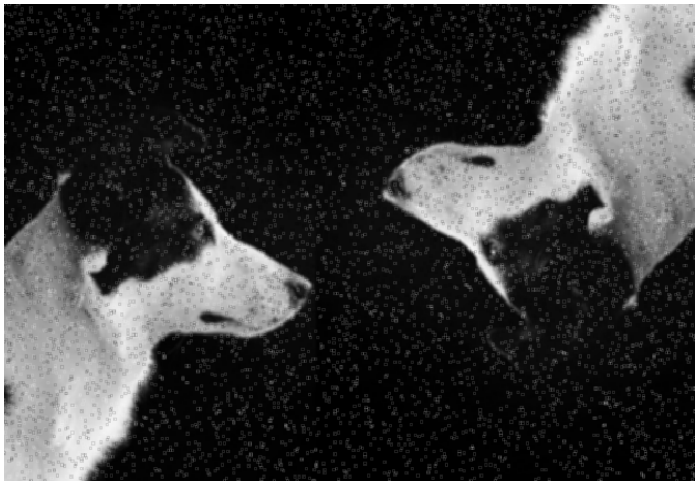


תמונות עם רעש *salt&pepper* לפני ואחרי סינון (גודל מסנן k=3):

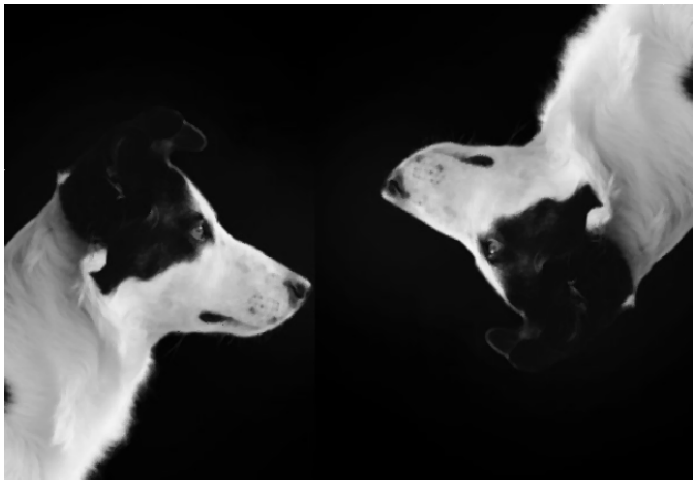
Salt&pepper noised image



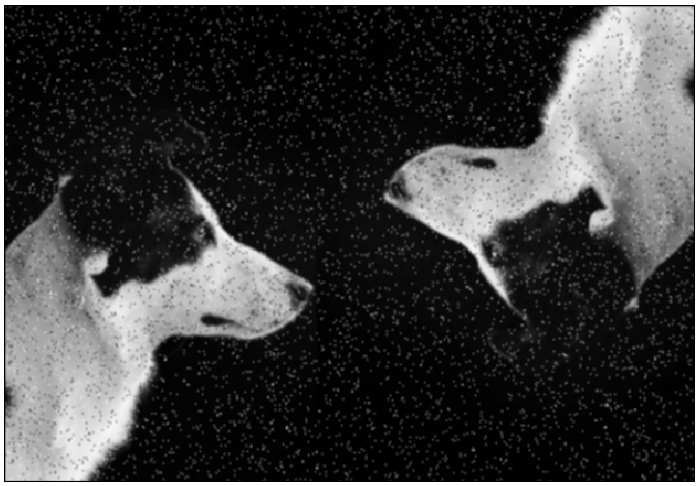
Salt&pepper noise- mean filter k=3



Salt&pepper- median filter k=3

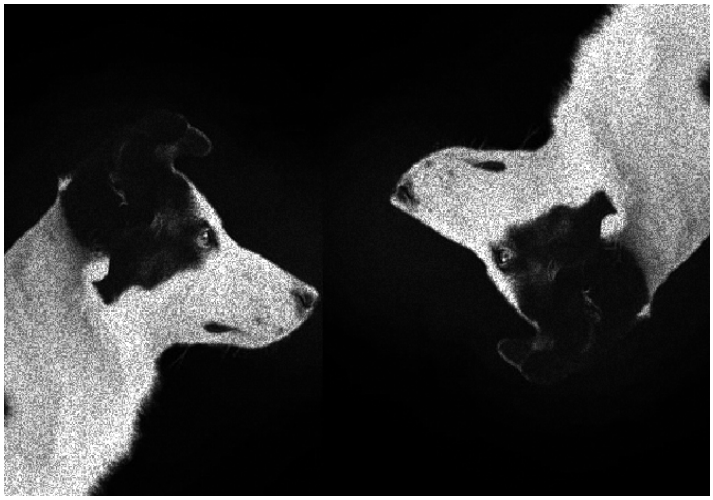


Salt&pepper noise- gaussian filter k=3,sigma=1

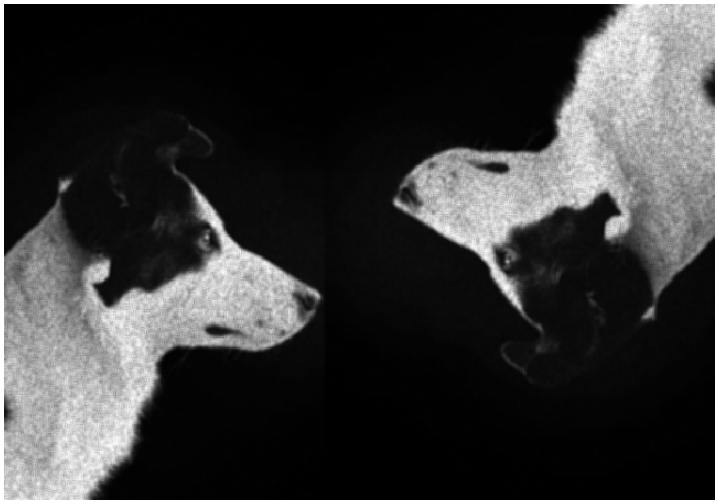


תמונות עם רעש *speckle* לפני ואחרי סינון (גודל מסנן k=3):

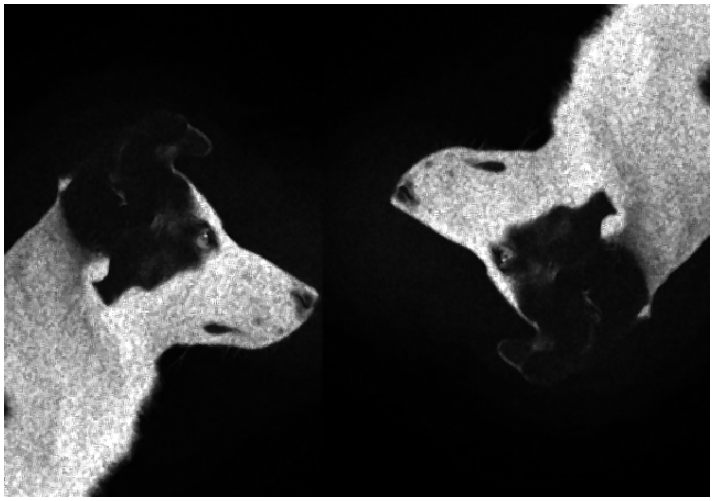
Speckle noised image



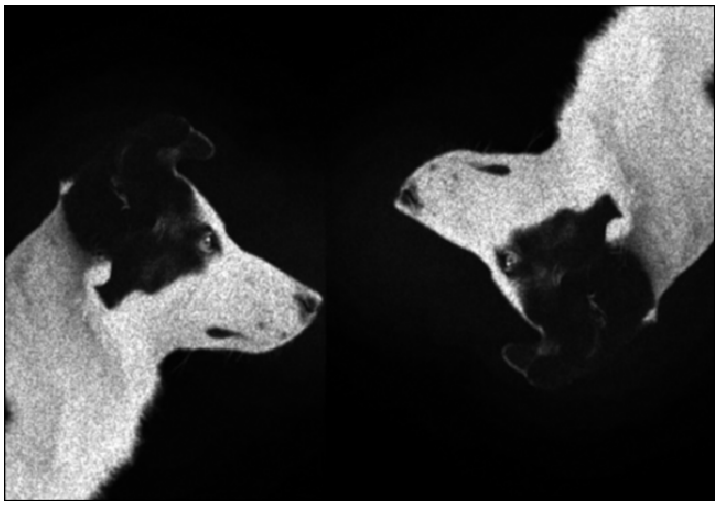
Speckle noise- mean filter k=3



Speckle- median filter k=3

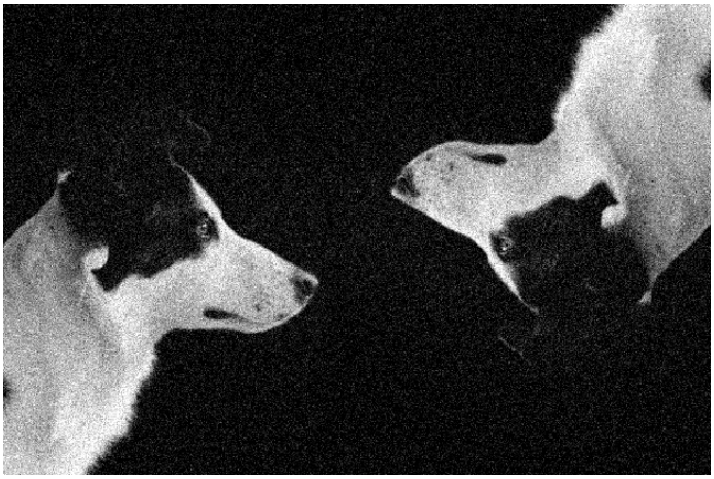


Speckle noise- gaussian filter k=3,sigma=1

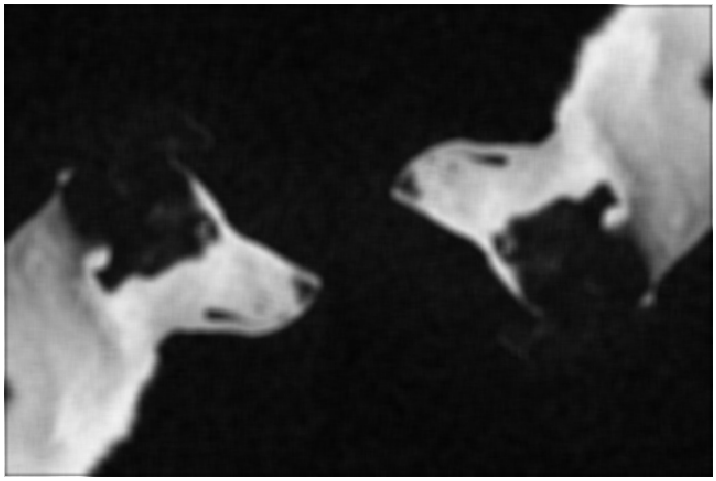


תמונות עם רעש גאוסי לפני ואחרי סינון (גודל מסנן k=9):

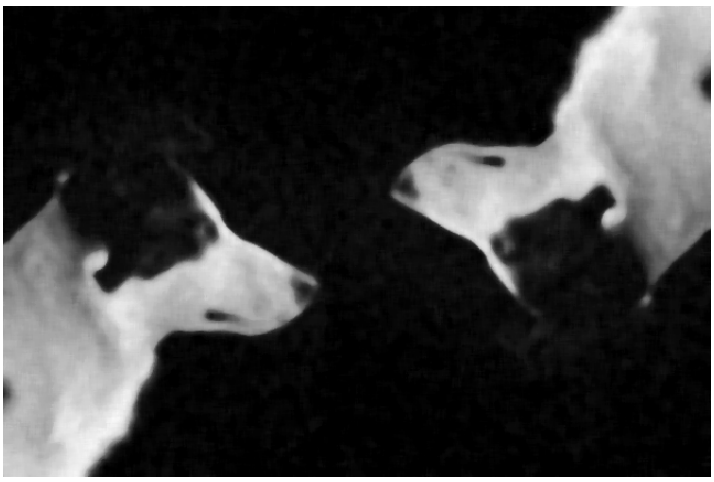
Gaussian noised image



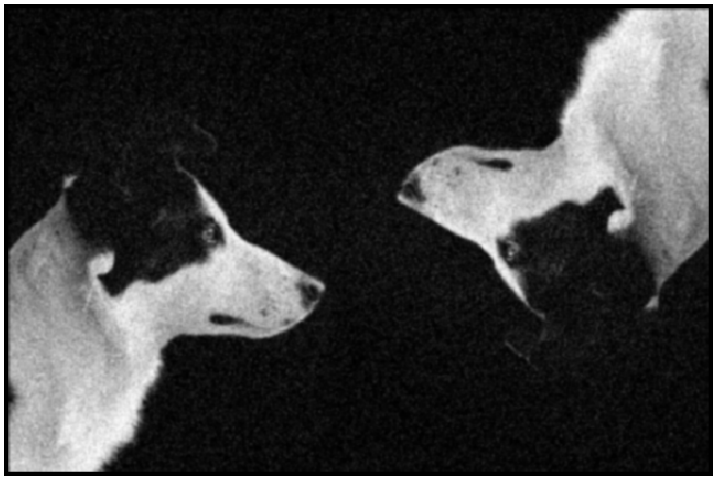
Gaussian noise- mean filter k=9



Gaussian noise- median filter k=9

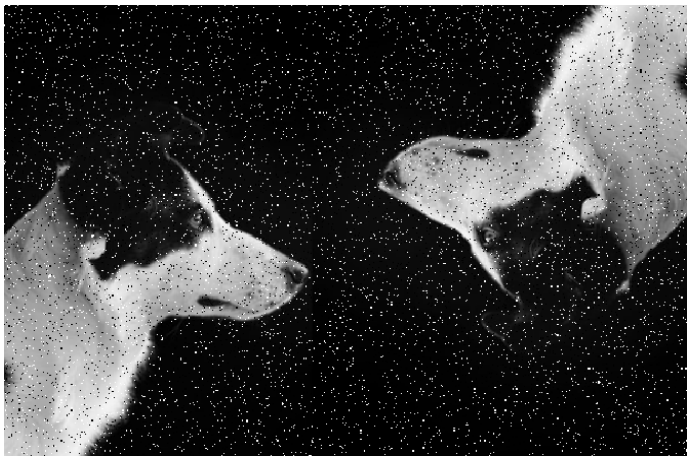


Gaussian noise- gaussian filter k=9,sigma=1

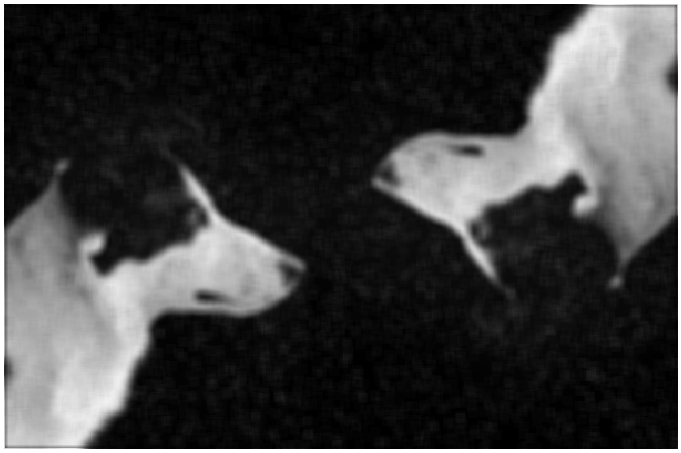


תמונות עם רעש *salt&pepper* לפני ואחרי סינון (גודל מסנן k=9):

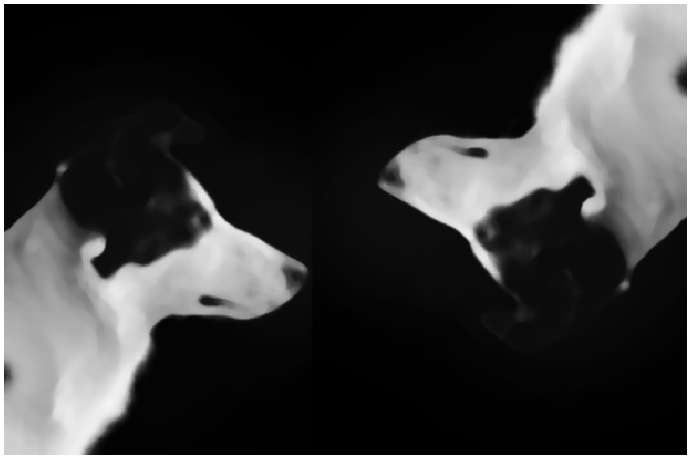
Salt&pepper noised image



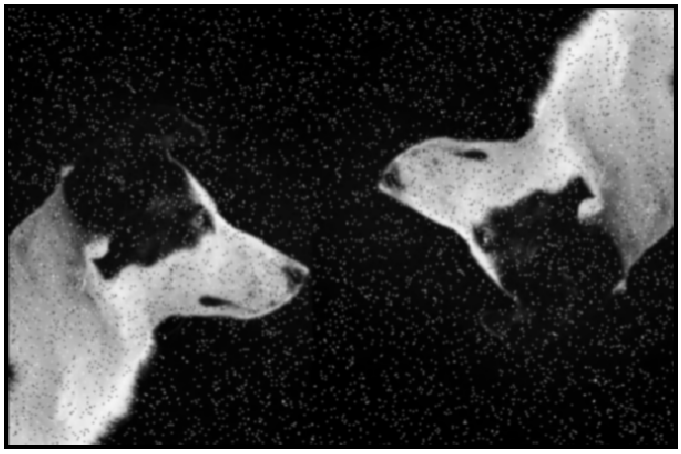
Salt&pepper noise- mean filter k=9



Salt&pepper- median filter k=9

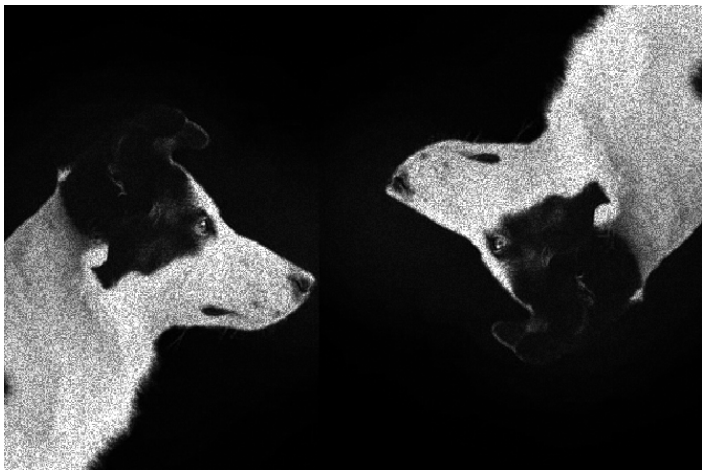


Salt&pepper noise- gaussian filter k=9,sigma=1

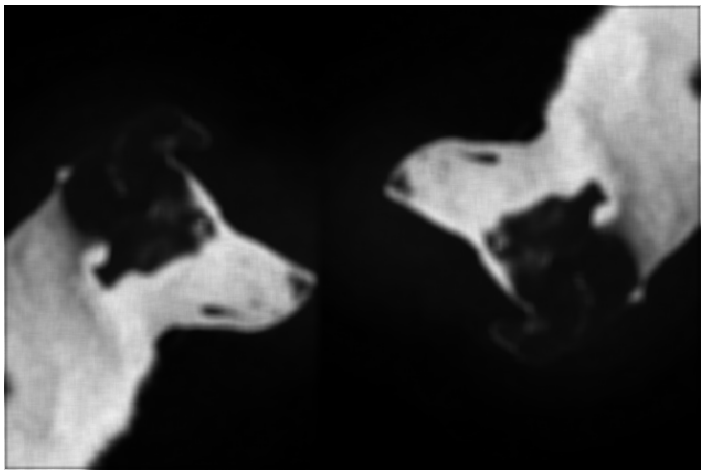


תמונות עם רעש *speckle* לפני ואחרי סינון (גודל מסנן k=9):

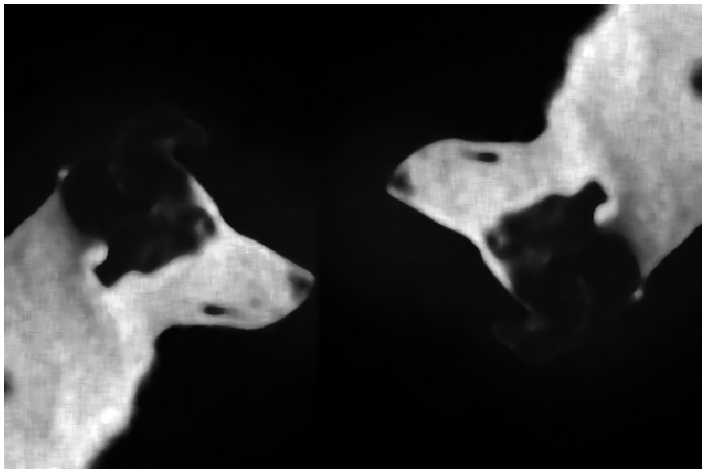
Speckle noised image



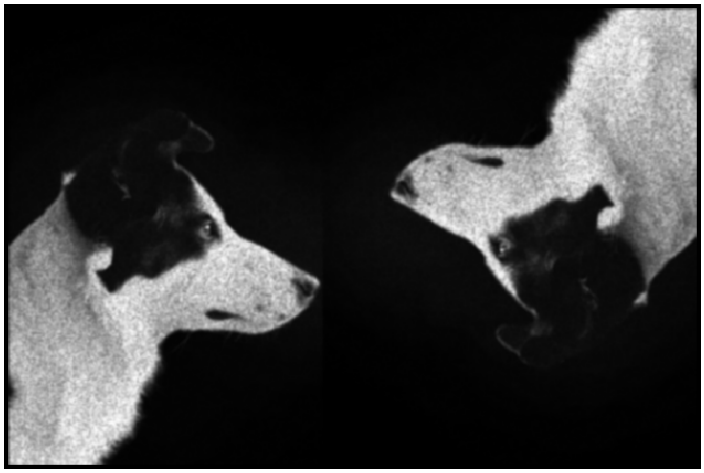
Speckle noise- mean filter k=9



Speckle- median filter k=9

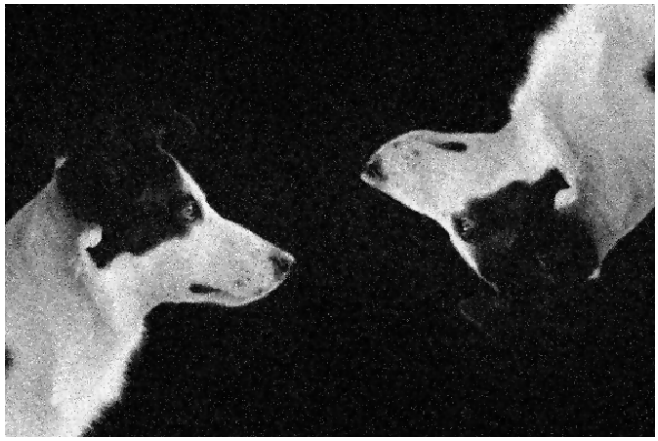


Speckle noise- gaussian filter k=9,sigma=1

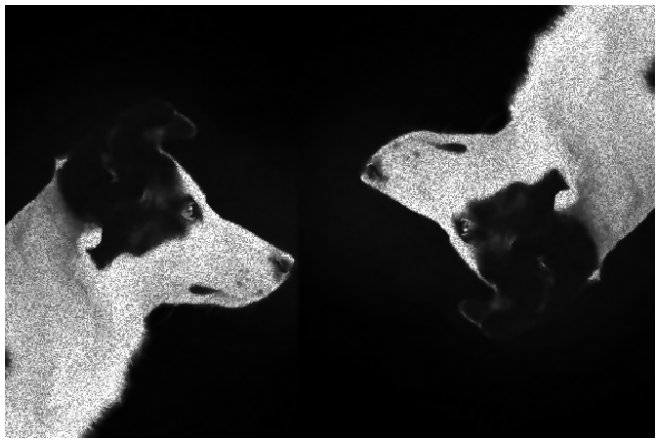


תמונות לאחר סינון על ידי מסנן Anisotropic Diffusion לכל אחד מסוגי הרעשים:

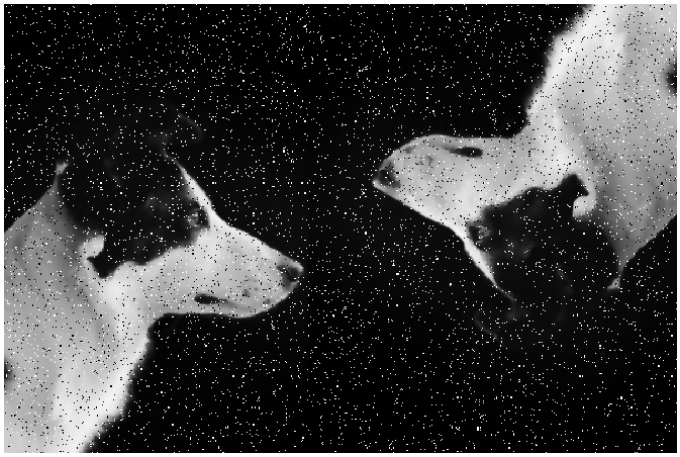
Gaussian noise- Anisotropic Difiusion filter



Speckle- Anisotropic Difiusion filter



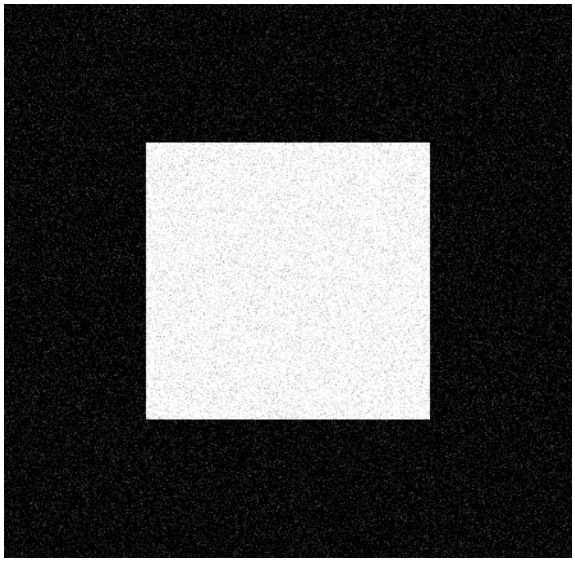
Salt&pepper noise- Anisotropic Difiusion filter



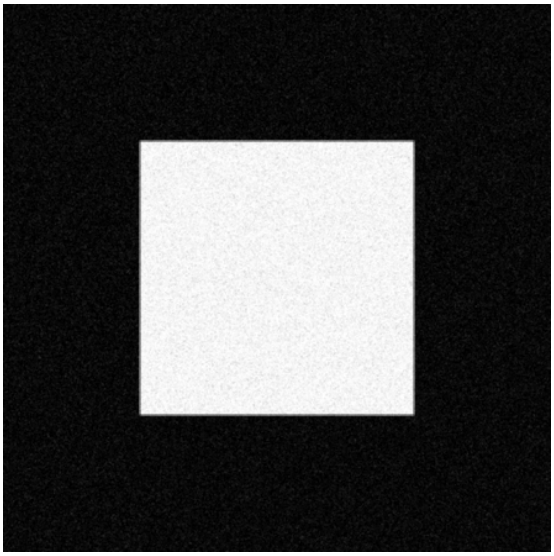
4. נחזור חלילה על הסעיף הקודם עבור התמונה *'square.jpg'* ונציג את התמונות המתקבלות:

תמונות עם רעש גאוסי לפני ואחרי סינון (גודל מסנן k=3):

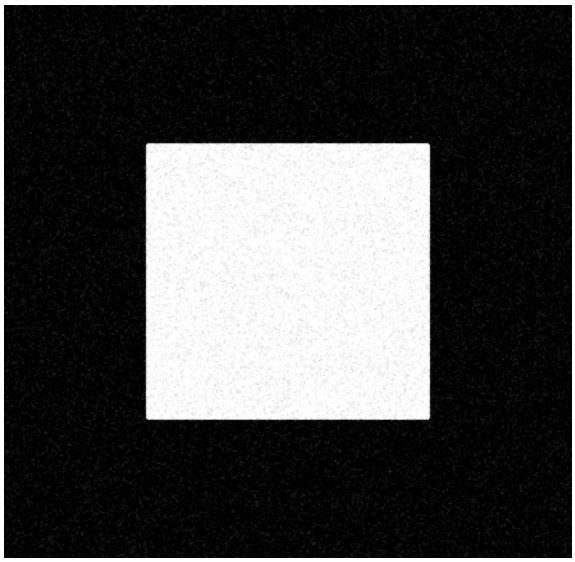
Gaussian noised image



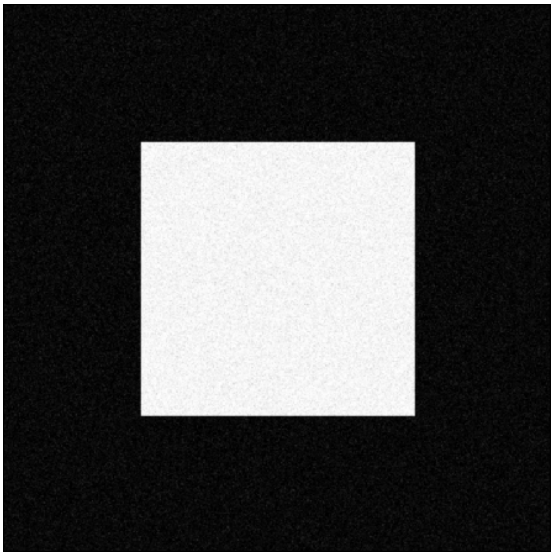
Gaussian noise- mean filter k=3



Gaussian noise- median filter k=3



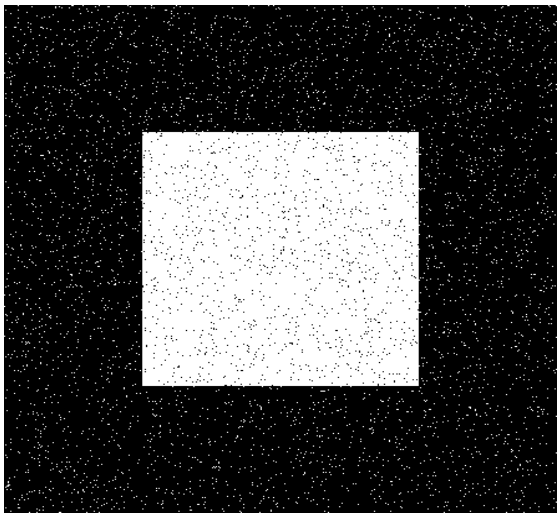
Gaussian noise- gaussian filter k=3,sigma=1



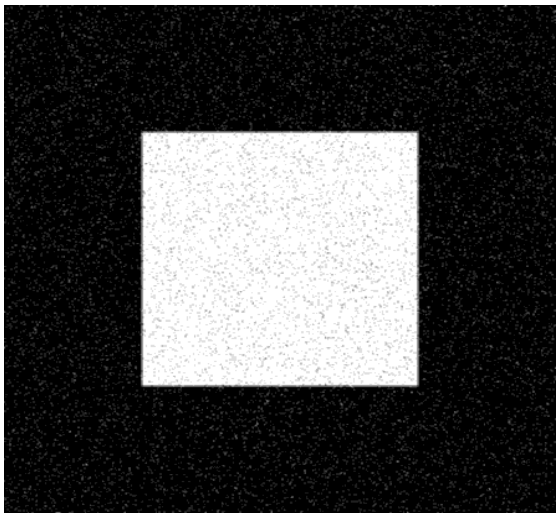
נשים לב כי עבור תמונה המורעשת ברעש גאוסי, המסנן הגאוסי ומסנן החציון עושים עבודה טובה יותר בניקוי הרעש. בתמונה של המסנן הממוצע השפות של התמונה יותר מטושטשות והרעש ניכר יותר לעין, זאת מאחר שרוב הפיקסלים בתמונה מורעשים ולכן מיצוע של הפיקסלים לא ימנע מהרעש. התמונה החדה ביותר שמתקבלת היא על ידי המסנן הגאוסי.

תמונות עם רעש *salt&pepper* לפני ואחרי סינון (גודל מסנן k=3):

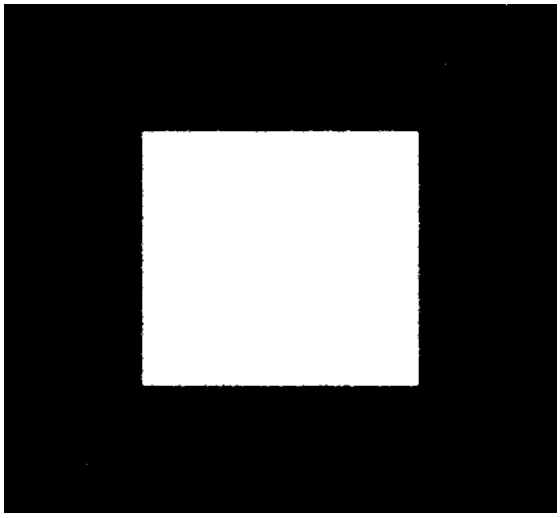
Salt&pepper noised image



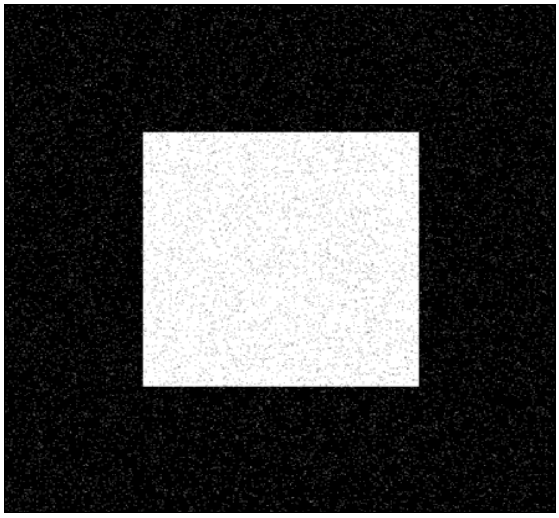
Salt&pepper noise- mean filter k=3



Salt&pepper- median filter k=3



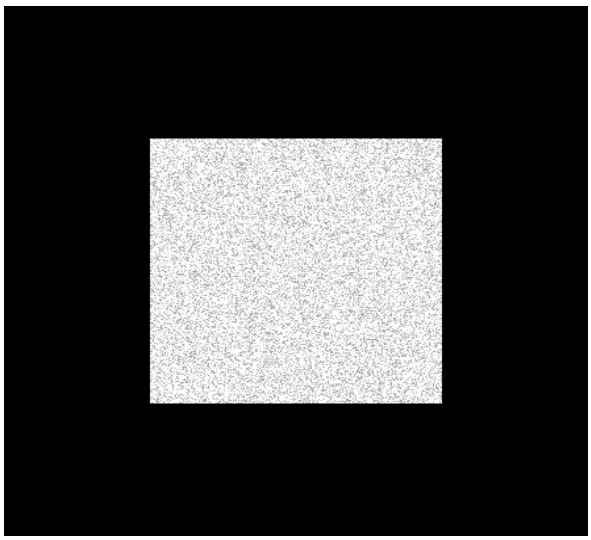
Salt&pepper noise- gaussian filter k=3,sigma=1



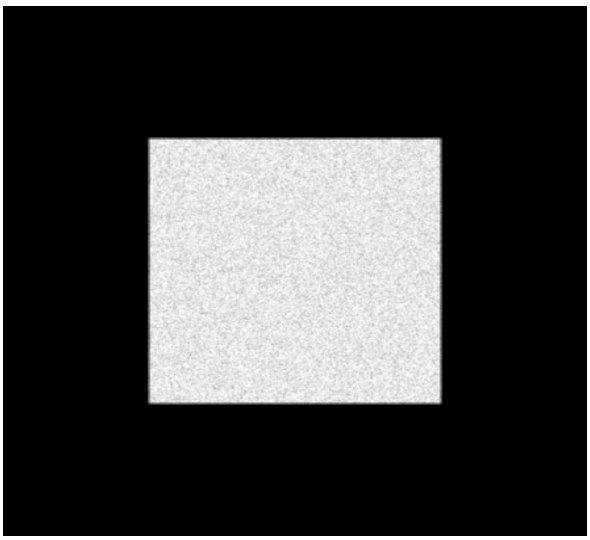
בתמונה עם רעש מסוג זה, שלושת המסננים הצליחו להפחית בעוצמת הרעש.
מסנן החציון עושה את העבודה הטובה ביותר בניקוי הרעש, מאחר שהחציון מאפשר להימנע מלקחת פיקסלים שערכם הפוך מזה של התמונה המקורית. אך הבעייתיות מופיעה בשפות של התמונה, שם יש זליגות של צבע שחור ללבן והפוך.
המסנן הגאוסי מנקה בצורה פחות טובה את הרעשים מאחר שהפיקסל המרכזי מקבל את המשקל הגדול ביותר ולכן אם הוא מורעש, סינון הרעש לא יהיה טוב מספיק מאחר ששכניו מקבלים משקל נמוך יותר, לעומת המסנן הממוצע שמושפע משכניו בצורה שווה.

תמונות עם רעש *speckle* לפני ואחרי סינון (גודל מסנן k=3):

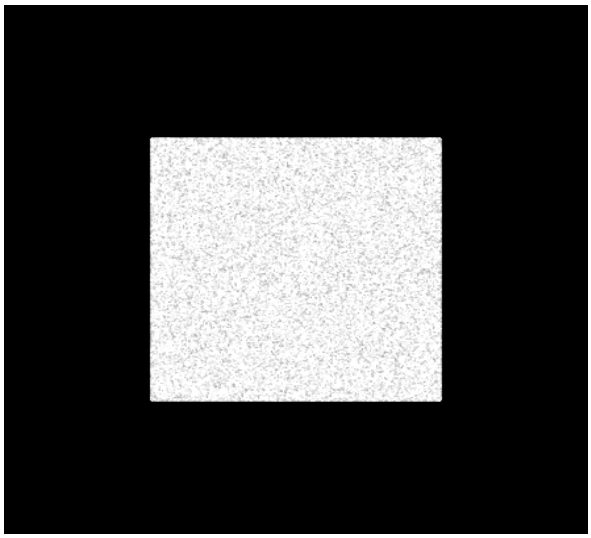
Speckle noised image



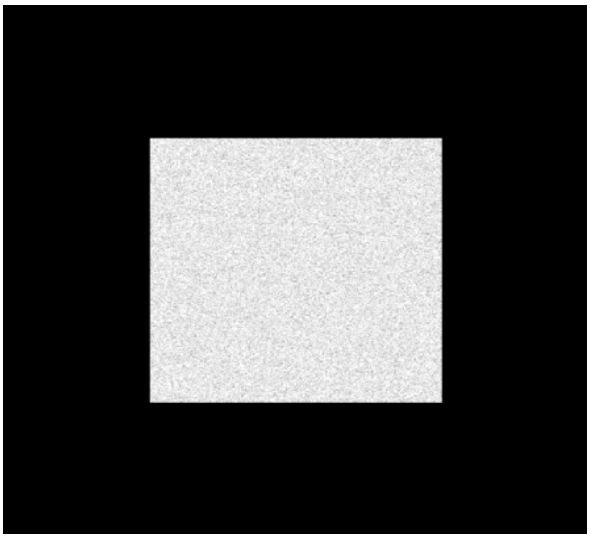
Speckle noise- mean filter k=3



Speckle- median filter k=3



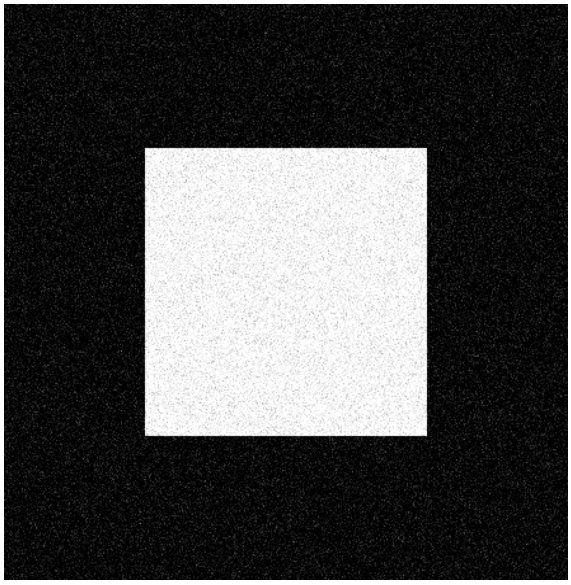
Speckle noise- gaussian filter k=3,sigma=1



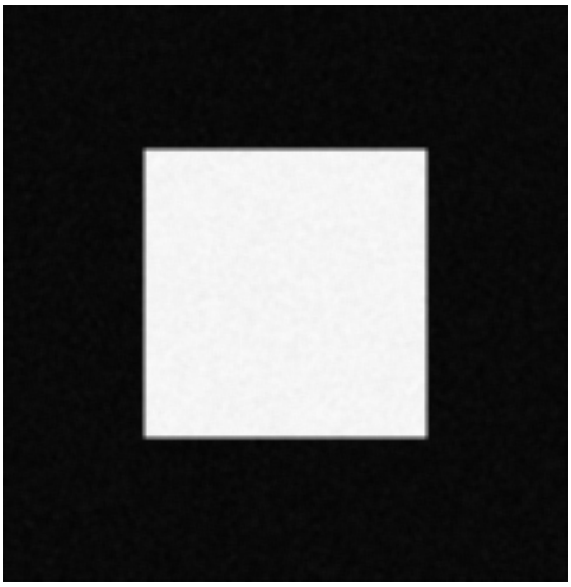
כאן נראה כי המסנן שעושה את העבודה הגרועה ביותר הוא מסנן הממוצע שבו השפות גם מטושטשות. מסנן החציון והמסנן הגאוסי עושים עבודה קצת טובה יותר והם בעלי שפות יותר חדות אך גם הם לא מבצעים ניקוי רעשים טוב במיוחד. המסנן הגאוסי גרם לתמונה להיות חלקה יותר לעומת מסנן החציון.

תמונות עם רעש גאוסי לפני ואחרי סינון (גודל מסנן k=9):

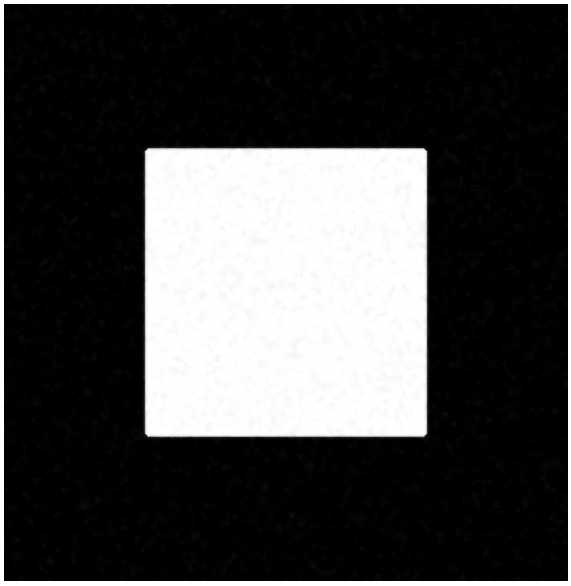
Gaussian noised image



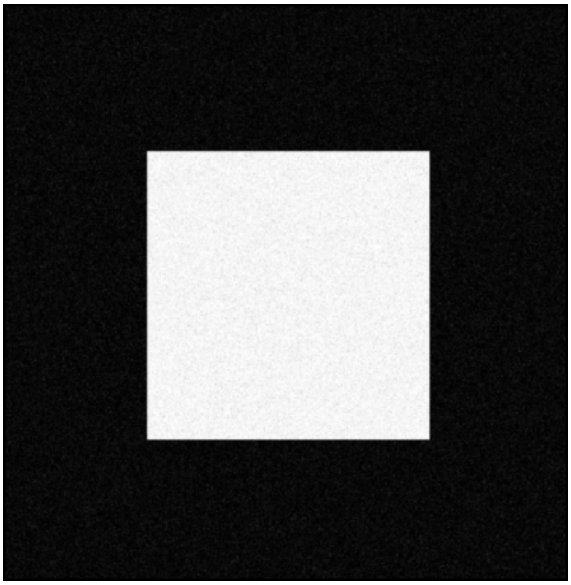
Gaussian noise- mean filter k=9



Gaussian noise- median filter k=9



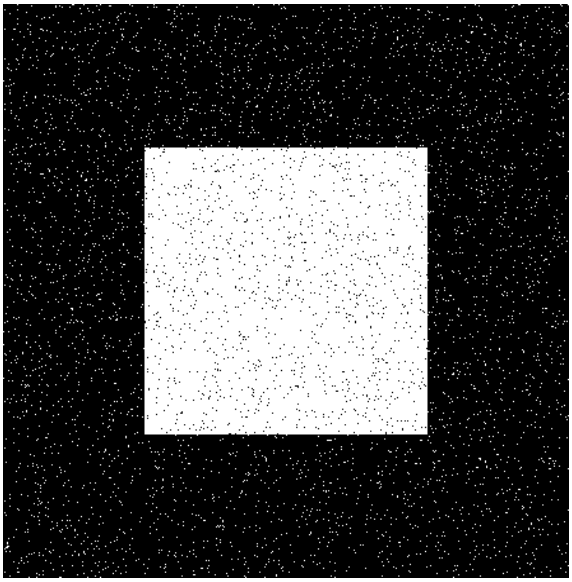
Gaussian noise- gaussian filter k=9,sigma=1



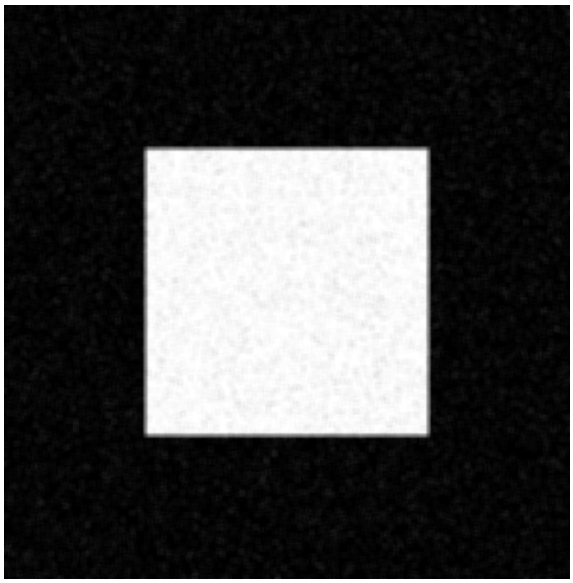
כאן ניקוי הרעשים נעשה בצורה טובה יותר מהמסננים הקטנים יותר. נשים לב שהמסנן הממוצע גורם לשפות של התמונה להיטשטש, ואילו המסנן החציון גורם לשפות להתעגל. במסנן הגאוסי עדיין ניתן לראות במקצת את הרעש אך פחות לעומת המסנן הגאוסי הקטן יותר.
באופן כללי, ניקוי הרעשים נעשה בצורה הטובה ביותר על ידי מסנן החציון.

תמונות עם רעש *salt&pepper* לפני ואחרי סינון (גודל מסנן k=9):

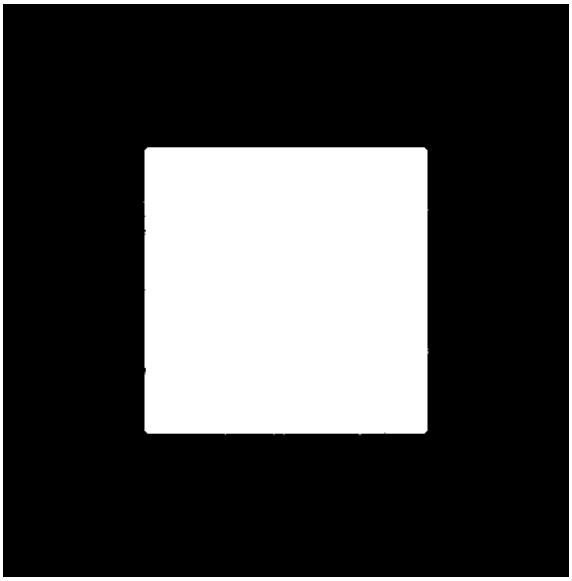
Salt&pepper noised image



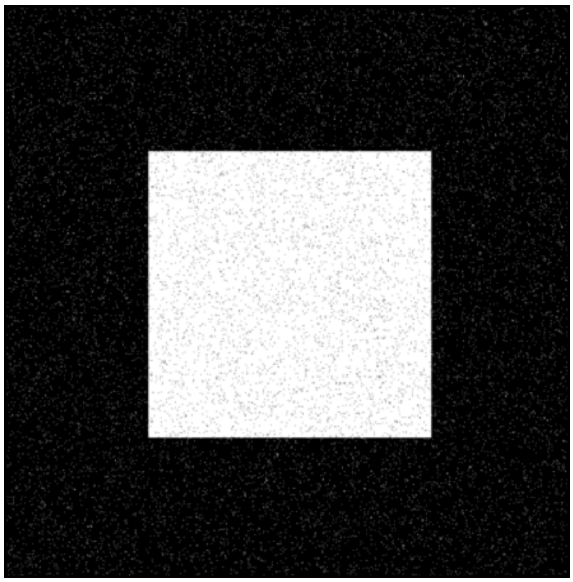
Salt&pepper noise- mean filter k=9



Salt&pepper- median filter k=9



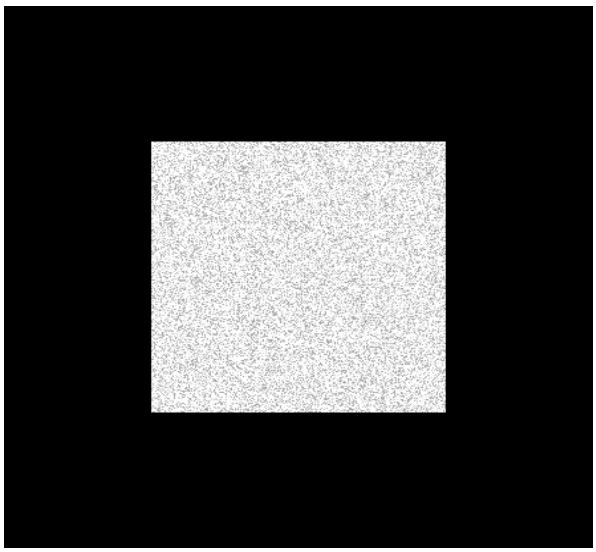
Salt&pepper noise- gaussian filter k=9,sigma=1



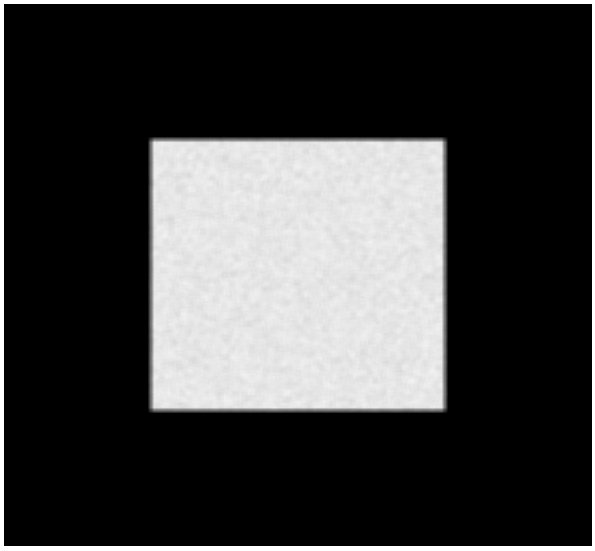
גם כאן ניקוי הרעשים נעשה בצורה הטובה ביותר על ידי מסנן החציון אך הבעייתיות היא בשפות של התמונה שמתעגלות במקצת (אך לעומת המסנן הקטן יותר, כאן זה פחות מורגש) .
ולגבי המסנן הגאוסי ומסנן ממוצע, גם כאן ניקוי הרעשים לא נעשה בצורה טובה והשפות של התמונה עם המסנן הממוצע מטושטשים.

תמונות עם רעש *speckle* לפני ואחרי סינון (גודל מסנן k=9):

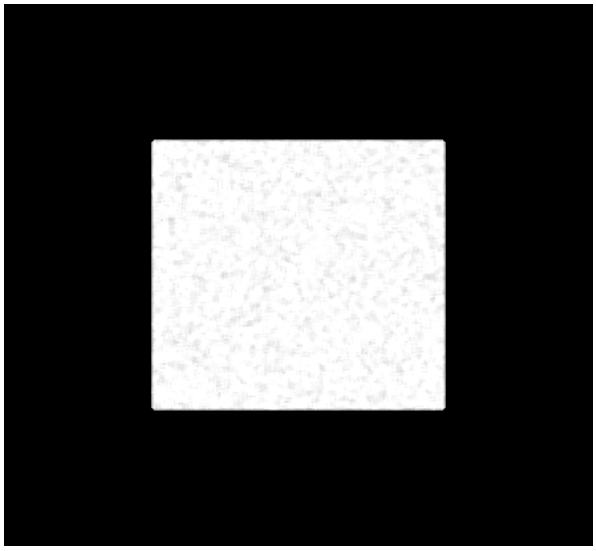
Speckle noised image



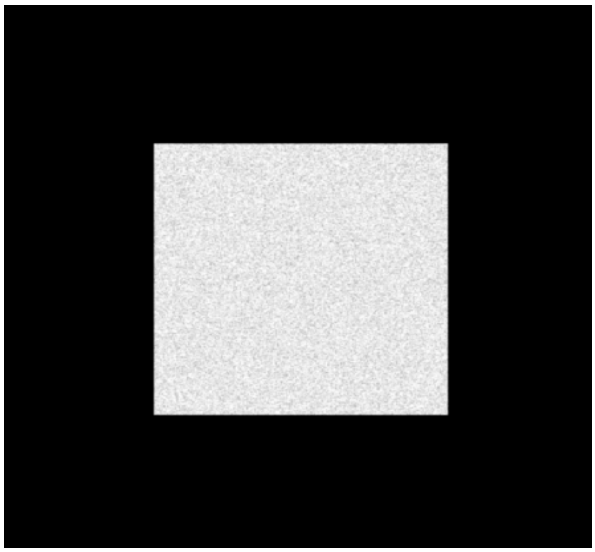
Speckle noise- mean filter k=9



Speckle- median filter k=9



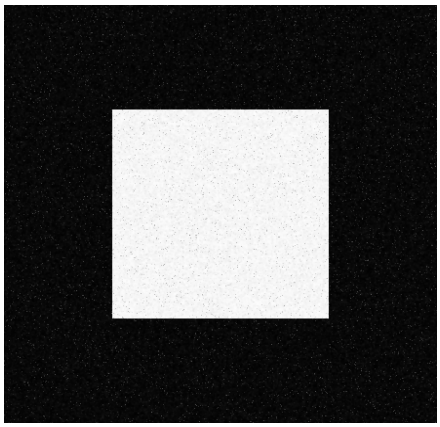
Speckle noise- gaussian filter k=9,sigma=1



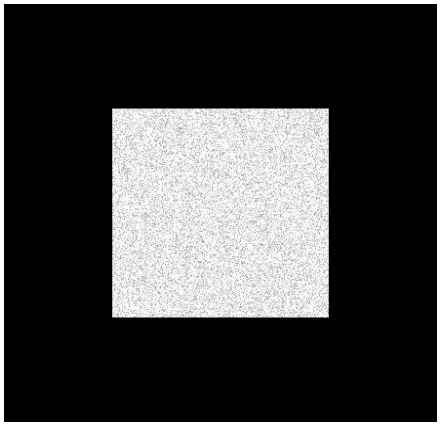
שוב, מסנן החציון עשה עבודת ניקיון טובה יותר מיתר המסננים , אך גם פה השפות קצת מתעגלות והתמונה מעט מרוחה (אם כי, בצורה פחות ניכרת לעין) . המסנן הגאוסי עשה עבודה טובה יותר לעומת המסנן הממוצע שבו השפות מטושטשות ואילו התמונה של המסנן הגאוסי הרבה יותר חדה.

תמונות לאחר סינון על ידי מסנן Anisotropic Diffusion לכל אחד מסוגי הרעשים:

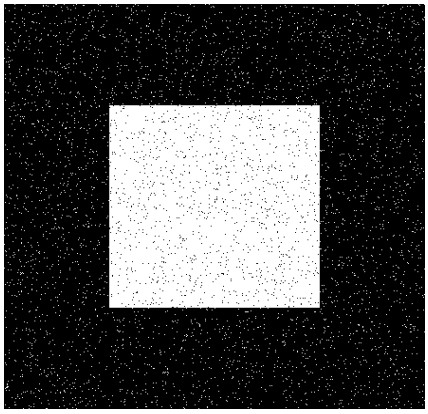
Gaussian noise- Anisotropic Difiusion filter



Speckle- Anisotropic Difiusion filter



Salt&pepper noise- Anisotropic Difiusion filter

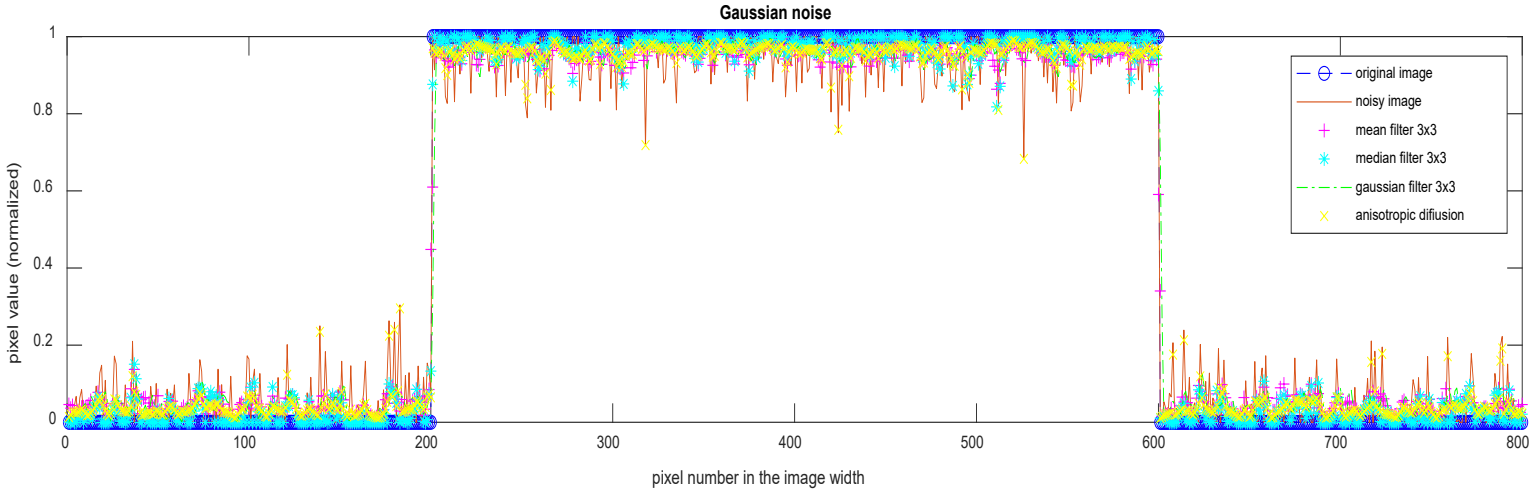


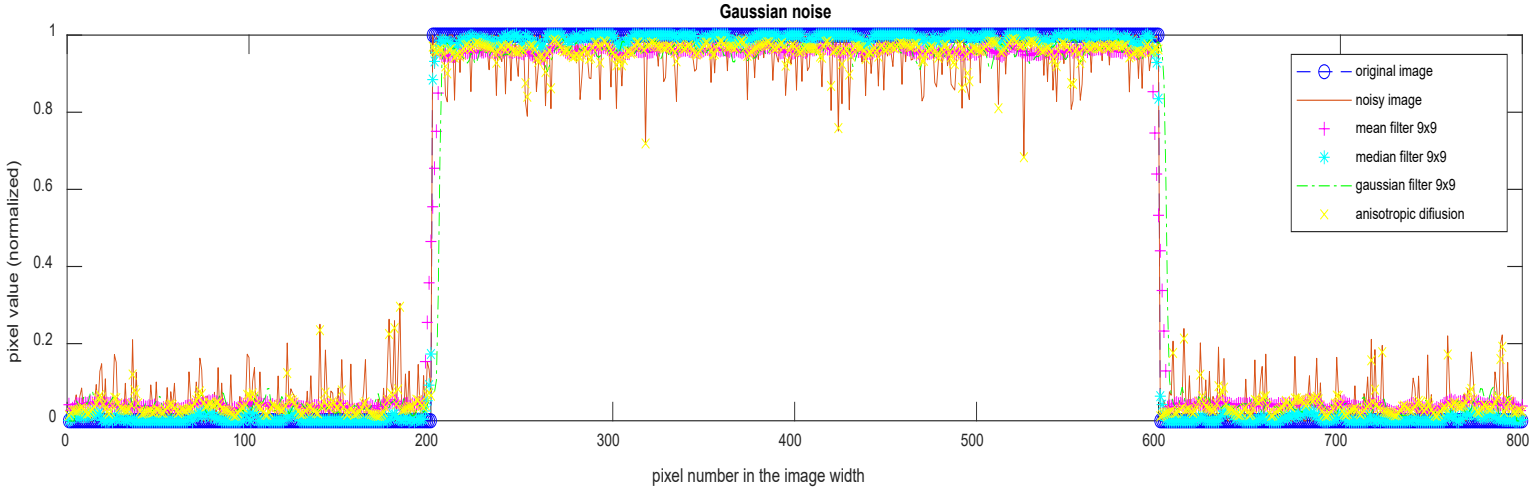
נשים לב כי מסנן זה לא מבצע ניקוי רעשים טוב לכל אחד מסוגי הרעשים. ניתן להסביר זאת בכך שמסנן זה מנסה לשמר את החלקים המשמעותיים של התמונה ולכן לצורך העניין עבור רעש *salt&pepper* קשה יותר להבחין בין רעש לתמונה עצמה מאחר שהתמונה סימטרית ובעלת שני צבעים בלבד. לכן ישנו קושי להבחין בין הרעש לבין התמונה.

נציג את הגרפים שיעזרו לנו להבין את ההשפעה של כל פילטר על התמונות המורעשות:

בגרפים הבאים נציג את ההשפעה של כל סוג מסנן על התמונה המורעשת בסוג רעש שונה.
בכל גרף מוצגים ערכי הפיקסלים המנורמלים שנמצאים באמצע רוחב התמונה (לקחנו חתך של התמונה המכיל את כל הצבעים, הרי מדובר בתמונה סימטרית לכל הצדדים). נרצה לראות איזה מבין הפילטרים מבצע סינון רעשים בצורה טובה יותר, כלומר להבחין איזה מהמסננים מביאים חזרה את הפיקסלים לערכים 0 ו-1 בצורה נקייה יותר.

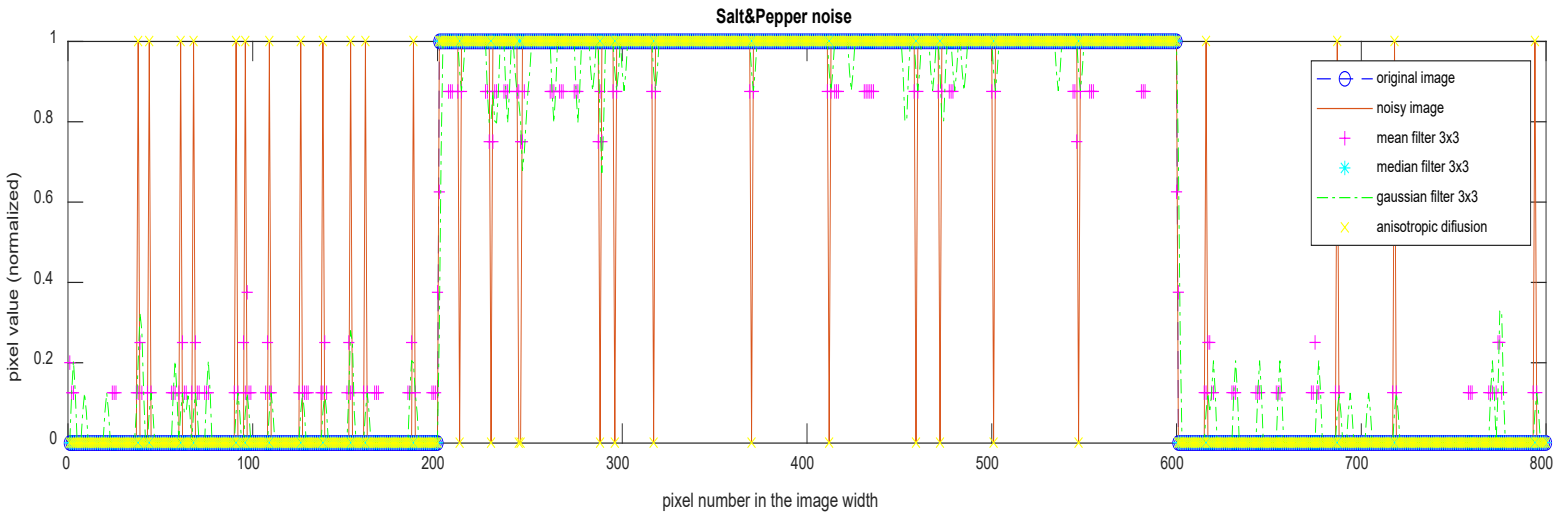
נתחיל מהתמונה המורעשת ברעש גאוסי:
בגרף העליון אנו רואים את ערכי התמונה המקורית שהם 0 או 1, התמונה המורעשת וערכי הפיקסלים של כל אחת מהתמונות המסוננות על ידי הפילטרים בגול $3x3$.
בגרף התחתון אנו רואים את אותו הדבר עבור פילטרים בגודל $9x9$.

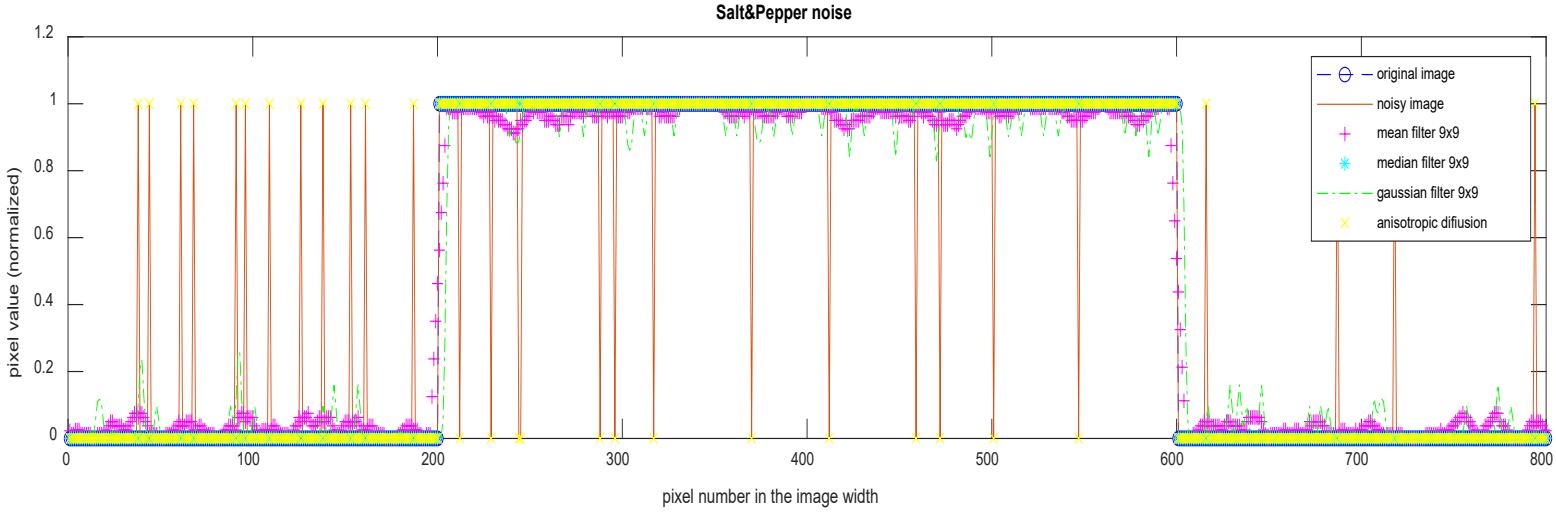




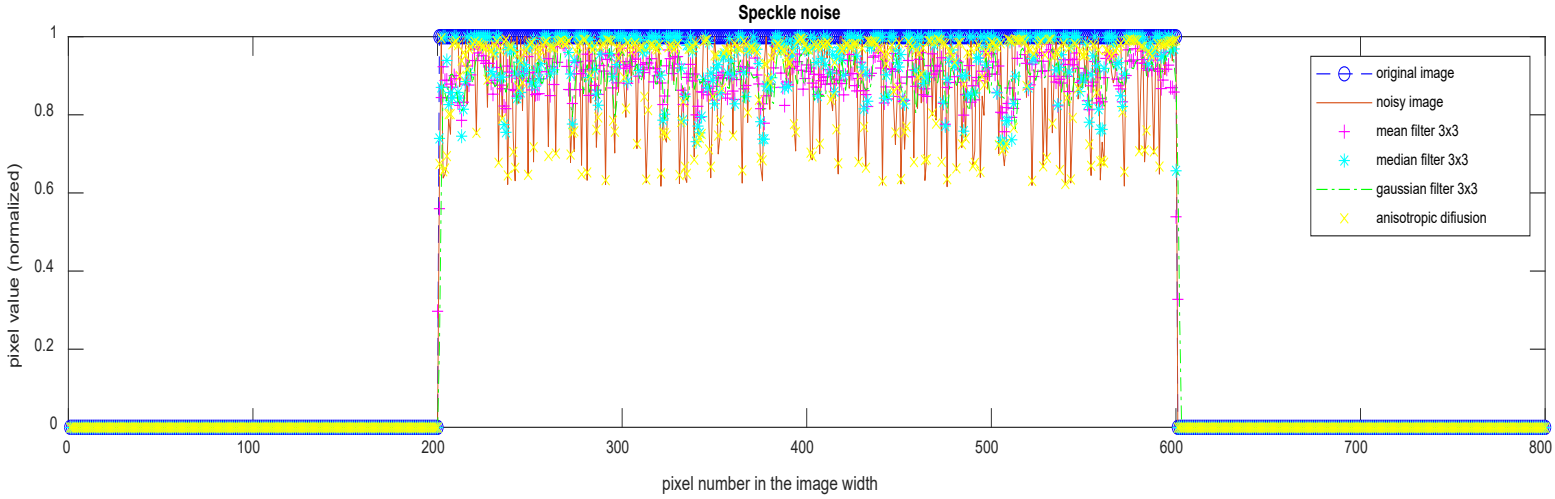
במבט ראשון, לא ניתן לראות הבדל מהותי בין הגרפים, אך אם נשים לב, נוכל להבחין כי ערכי הפיקסלים של התמונה המסוננת על ידי פילטר חציון בגודל $9x9$ קרובים יותר ל-0 ול-1 לעומת התמונה המסוננת על ידי פילטר חציון בגודל $3x3$, מה שמצביע על כך שהמסנן חציון בגודל $9x9$ עושה עבודה טובה יותר בסינון הרעשים. בנוסף נוכל להבחין כי בשני הגרפים, מסנן החציון מסנן רעשים בצורה טובה יותר מיתר המסננים, אך גם המסנן הגאוסי עושה עבודה לא רעה בסינון הרעשים, למרות שהשפות (מספר 200 ו-600) של התמונה לאחר פילטר גאוסי מטושטשות.

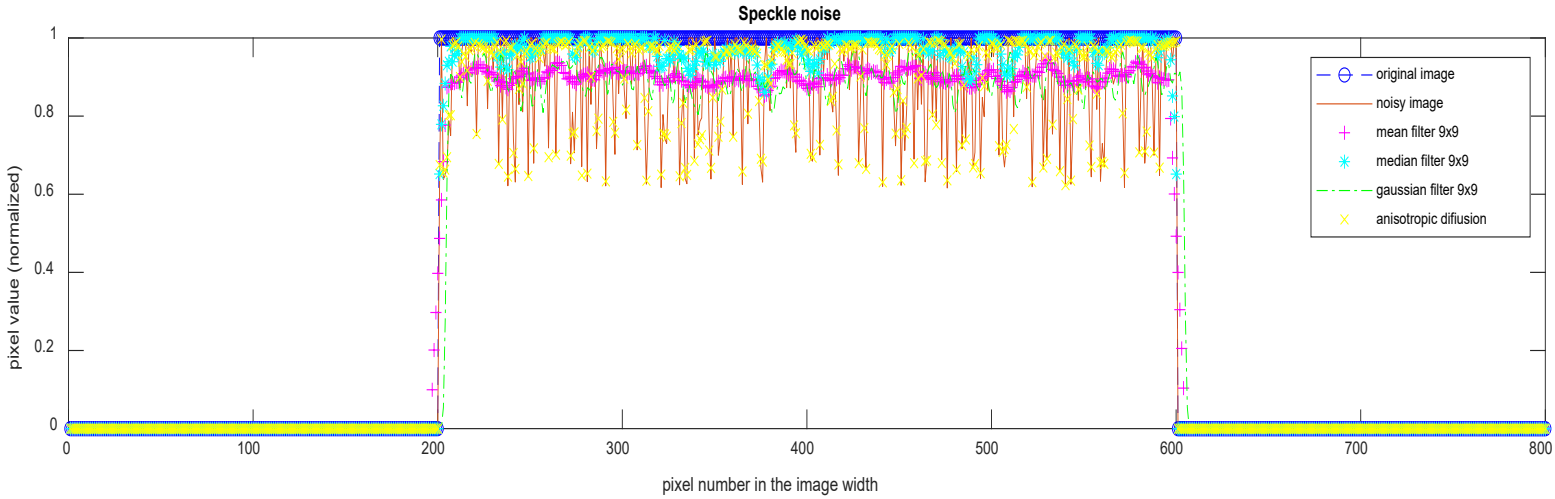
נעבור לרעש *salt&pepper* :
כאן הרעש הרבה יותר בולט לעין. כאן ברור כי המסנן הגאוסי, המסנן הממוצע והמסנן Anisotropic Diffusion ביצעו סינון רעשים הרבה פחות טוב מסינון הרעשים על ידי מסנן החציון.
המסנן הגרוע ביותר של רעש זה הוא מסנן Anisotropic Diffusion.





נמשיך לסוג הרעש האחרון, רעש *speckle*:
נשים לב כי רעש זה בא לידי ביטוי באופן ניכר בעיקר בערך 1 (כלומר בצבע הלבן בתחום של הפיקסלים 200-600) . גם פה המסנן הגרוע ביותר הוא מסנן Anisotropic Diffusion.
נשים לב כי ערכי הפיקסלים של יתר המסננים נראים מכונסים יותר לערך 1 בגרך התחתון בו גודל הפילטרים הוא $9x9$, דבר המעיד על סינון רעשים טוב יותר.
בנוסף, ניתן לראות כי המסנן רעשים הטוב ביותר הוא פילטר החציון המציג סינון רעשים טוב יותר בגודל $9x9$.





לסיכום:
המסנן Anisotropic Diffusion מנקה בצורה הגרועה ביותר את הרעשים מכל הסוגים.
בנוסף, קיבלנו כי באופן כללי מסנן החציון מבצע עבודה טובה יותר בניקוי רעשים לעומת המסנן הממוצע. החיסרון שלו הוא שמקבלים תמונה מעט מרוחה ואילו בתמונות של המסנן הממוצע השפות מטושטשות.
מעבר לכך, היתרון הברור של המסנן הגאוסי הוא שלעומת כל היתר הוא ייצר את התמונה החדה ביותר אך עשה עבודה פחות טובה בניקוי הרעשים.