EECE 5639 COMPUTER VISION

PROJECT 1

Motion Detection Using Simple Image Filtering

Submitted By,
Sivashankar Venkatachalam
Dhanush Balakrishna

ABSTRACT

In this project, we have explored the simple technique for Motion Detection using background images in MATLAB. Motion Detection is one of the primary applications of Signal and image processing. Motion Detection involves analyzing an input image, whether it be image frames or video signals to detect the changes in the background, such as the movement of objects, people, etc. In other words, this process involves tracking the object and detecting its motion. For this, we have converted the series of multiple image frames into video input and applied the temporal derivative method to obtain the temporal gradient of the video frames. These video frames, in turn, capture the changes in threshold over frames. From the intensity of threshold variation between multiple frames, we have detected the motion on a stationary background. After this, we visualized the output of the detected motion as a contour around the moving object. For this project, we have experimented with various filters by applying different threshold and standard deviation values. The parameters and values we have adopted in this project are discussed briefly in the upcoming sections of this report. The project we have explored will be helpful in investigating further various applications of motion detection for analyzing and extracting information from visual signals to perform unique advanced computer vision tasks. Some real-time applications in today's world of this concept are object tracking and surveillance applications in autonomous systems.

ALGORITHM AND METHODOLOGY

To implement this project, the input dataset we have used is office chair data. The input data is a series of multiple image frames stored in the local folder of the system. We have implemented this concept using MATLAB software. Using MATLAB, we have converted the multiple image frames into a single video. Now the video acts as input for the other methods we have implemented. As given in the project plan document, we have read the input image frames of the video and converted them into a grayscale. After converting them into grayscale motion, we applied the masks to differentiate the areas where motion was detected. As a result, we have obtained two different portions in visuals where the black-masked part represents no motion, and the white-masked one represents the motion of the moving object. This is the basic principle involved in detecting the motion of the object. After detecting the motion, we masked it again with the original ones to get the desired output.

PARAMETERS

In this process, we have implemented some parameters to obtain the desired motion-detected frame as our output. As mentioned in the project requirements, we have calculated the temporal gradient using the abs diff function. The temporal gradient calculation is an essential step in this motion detection project as it provides the required information about the changes in the pixel varies with respect to the frame in the form of a threshold. To perform all these operations, we have used the processing_pipeline function in MATLAB programming. In general, this function can be helpful in performing various image processing operations such as filtering, detection, segmentation, and extraction. In our project, it has dramatically helped us to perform filtering and detection operations.

EXPERIMENTS AND OBSERVATIONS

The experiments have been carried out based on the requirements given in the question. Below section includes various images recorded during the execution of various experiments using MATLAB.As we discussed earlier, the below image shows the original motion-detected image after using the grayscale and masks. The headers have been clearly mentioned in the picture, and please find them for reference.

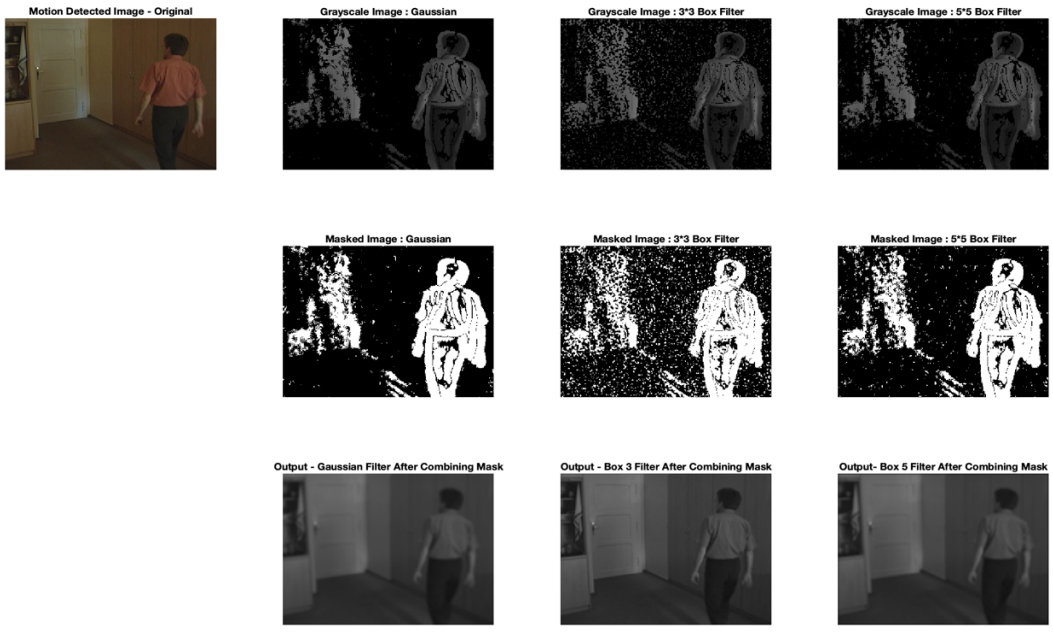


Figure 1.0

The first row in the above image depicts the grayscale image; the second row elucidates the masked image. Using the masked images, we can clearly differentiate the area where motion is detected and vice-versa. As we stated earlier, the motion detection area is represented in a white mask, and the area without motion is black. After detecting the motion, the masked image is combined with the original mask to visualize the desired output.

While implementing this, we have introduced different filters mentioned in the project definition. Those are the Gaussian filter, 3X3 box filter, and 5X5 box filter. From fig. 1.0, we can easily differentiate the smoothness of the image which we obtained after the application of different filters. The objective of employing box filters and gaussian filters is to improve video frame quality and minimize noise and artifacts. By enhancing the quality of the input frames, these filters can enhance the precision and dependability of motion detection algorithms.

Filters Analysis and Comparison

While comparing different filters, Gaussian filters are more frequently used, and this is because high-frequency noise may result in erroneous motion detections, which may affect the findings of motion detection. While reducing noise, Gaussian filters are made to protect the image's edges and details. On the other hand, box filters are used to average the image intensity values and lessen the impact of pixel-level noise. This can lessen the effect of minute changes in pixel values brought on by noise or changes in lighting, which can also result in misleading motion detections. Box filters are easier to use and can be computed more quickly than Gaussian filters, although they may result in blurrier images.

Both types of filters can be used together in a motion detection pipeline, with the box filter applied first to remove pixel-level noise and the Gaussian filter applied later to smooth the image and remove high-frequency noise. This can help to improve the accuracy and robustness of the motion detection algorithm and to reduce false detections.

Experimenting by user-defined ssigma and varying tsigma values

We have experimented by defining the standard deviation for Gaussian and varying tsigma values. Here we have captured two instances of tsigma with value 1 and 4. Below are the plots for the same.

Tsigma with value 1

Figure 1

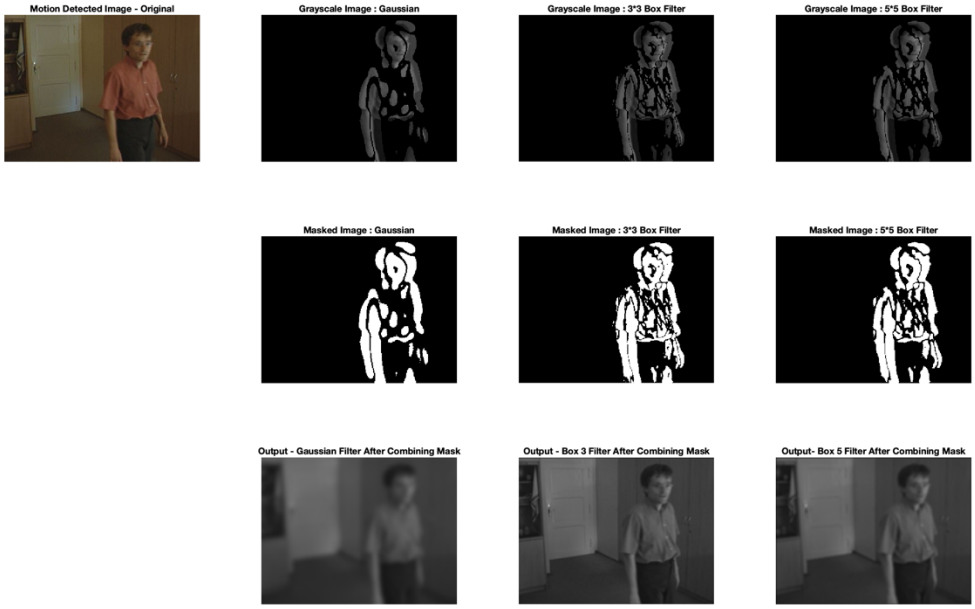


Figure 2

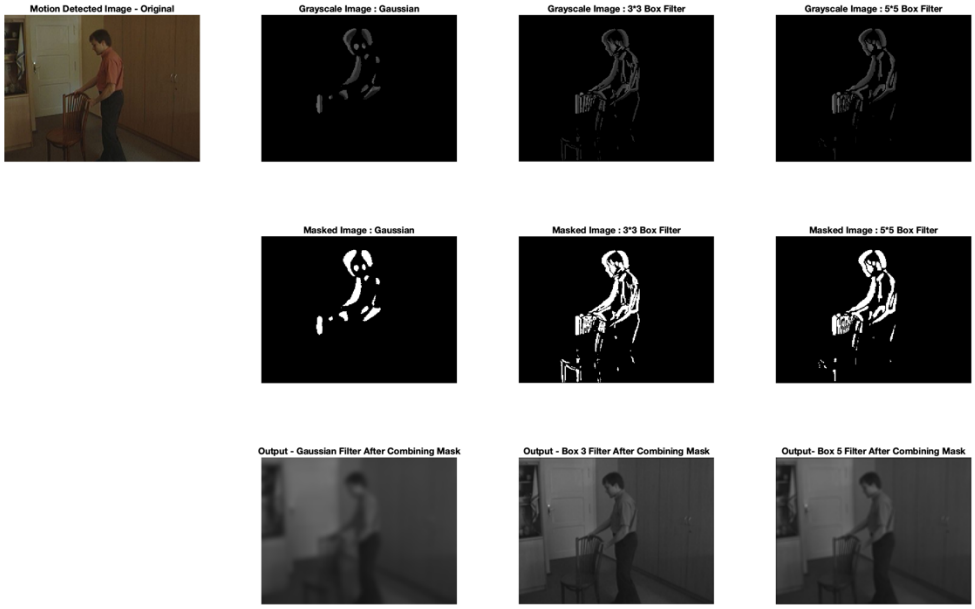
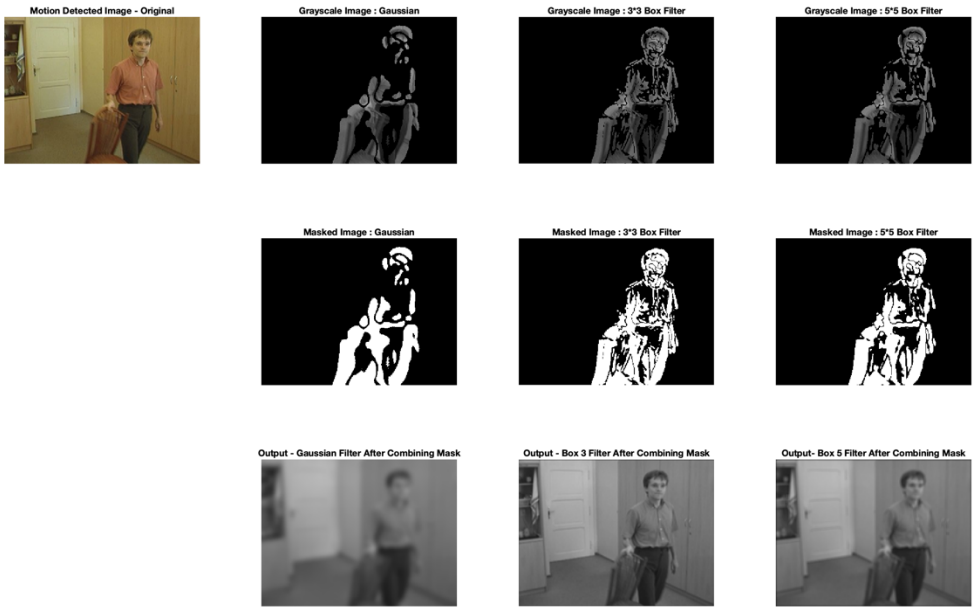
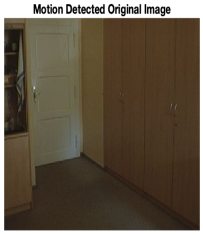
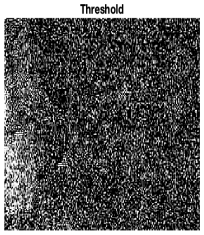
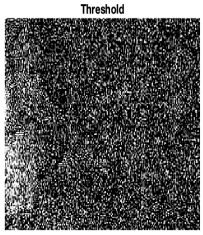


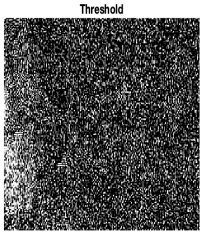
Figure 3

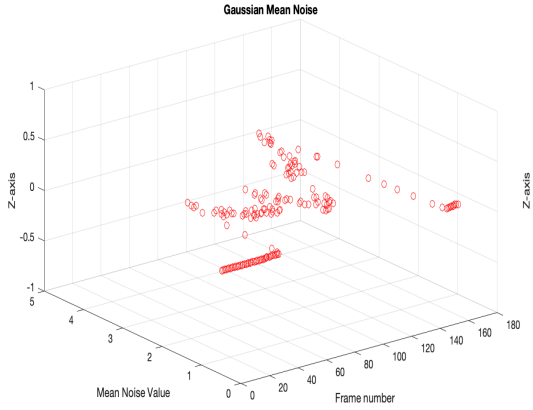


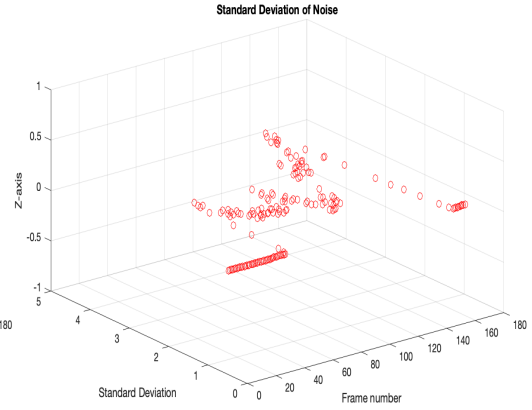

Motion Detected Original Image


Threshold


Threshold

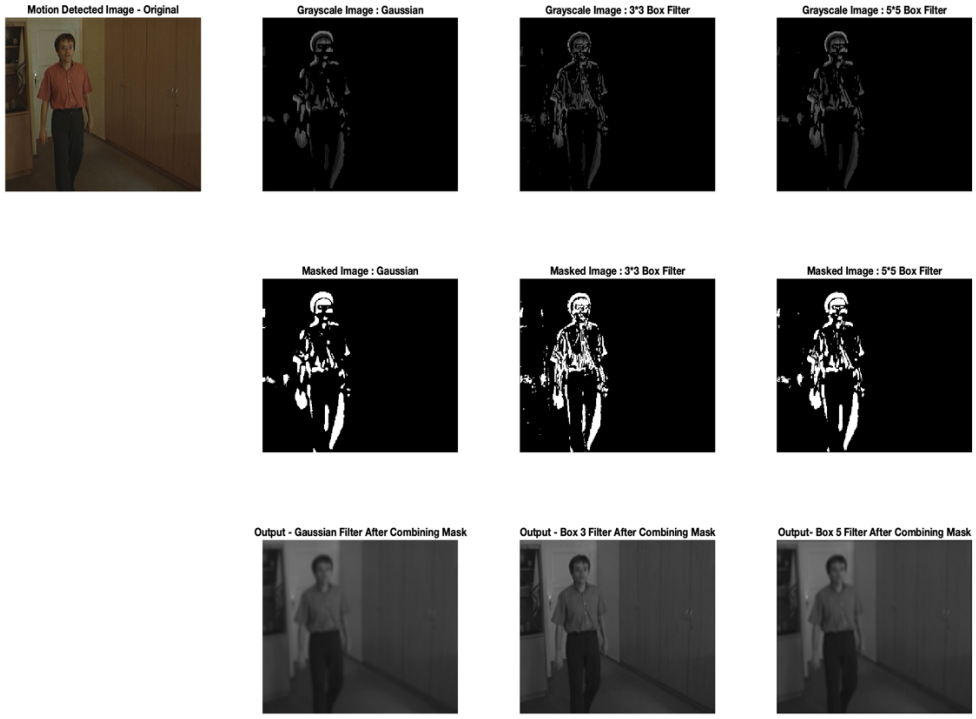

Threshold


Gaussian Mean Noise
Z-axis
Mean Noise Value
Frame number


Standard Deviation of Noise
Z-axis
Standard Deviation
Frame number

Tsigma with value 4

Figure 5



Motion Detected Image - Original

Grayscale Image : Gaussian

Grayscale Image : 3*3 Box Filter

Grayscale Image : 5*5 Box Filter

Masked Image : Gaussian

Masked Image : 3*3 Box Filter

Masked Image : 5*5 Box Filter

Output - Gaussian Filter After Combining Mask

Output - Box 3 Filter After Combining Mask

Output- Box 5 Filter After Combining Mask

Motion Detected Image - Original

Grayscale Image : Gaussian

Grayscale Image : 3*3 Box Filter

Grayscale Image : 5*5 Box Filter

Masked Image : Gaussian

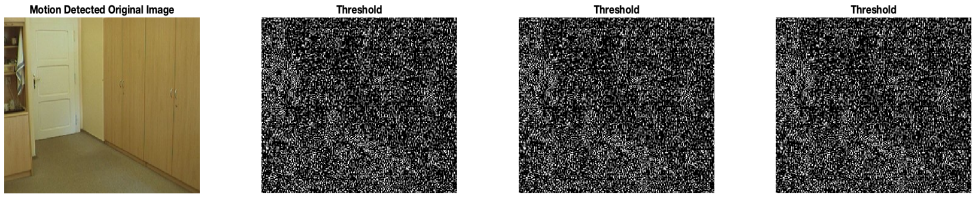
Masked Image : 3*3 Box Filter

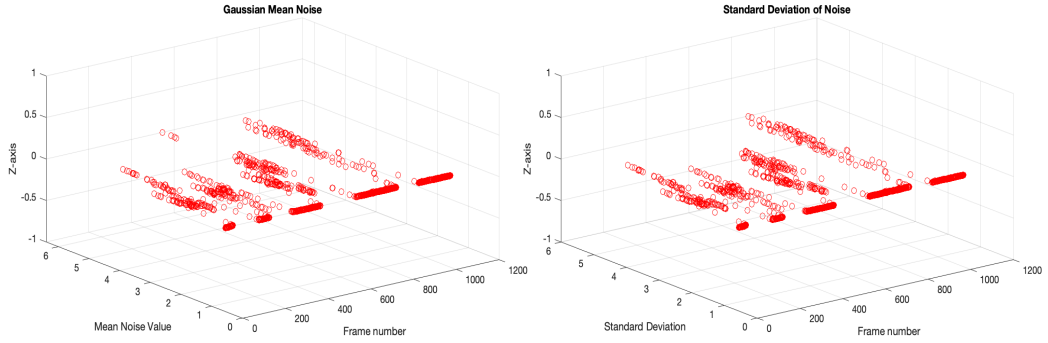
Masked Image : 5*5 Box Filter

Output - Gaussian Filter After Combining Mask

Output - Box 3 Filter After Combining Mask

Output- Box 5 Filter After Combining Mask





Yes, varying these will make the results better. On varying the tsigma and user-defined sigma for the temporal filters, we learned that the final image will be smoother and have more blurred edges as the Gaussian standard deviation rises. This is because a higher standard deviation number will spread the Gaussian distribution more widely, including more nearby pixels in the smoothing operation. As a result, the image's edges will be blurrier, and its finer features could be lost. As a result, it's crucial to choose a suitable value for the Gaussian standard deviation depending on the image's properties and the level of smoothing that is needed.

Experimentation by Varying Threshold Values

In our next experiment, we tried varying the threshold with various levels and recorded the output. Below are the images which we obtained while varying the threshold levels.Firstly, we have experimented with threshold value two. The below part includes a series of images that detects motion from the beginning to the end of the video that has been recorded here.

Threshold Value - 2

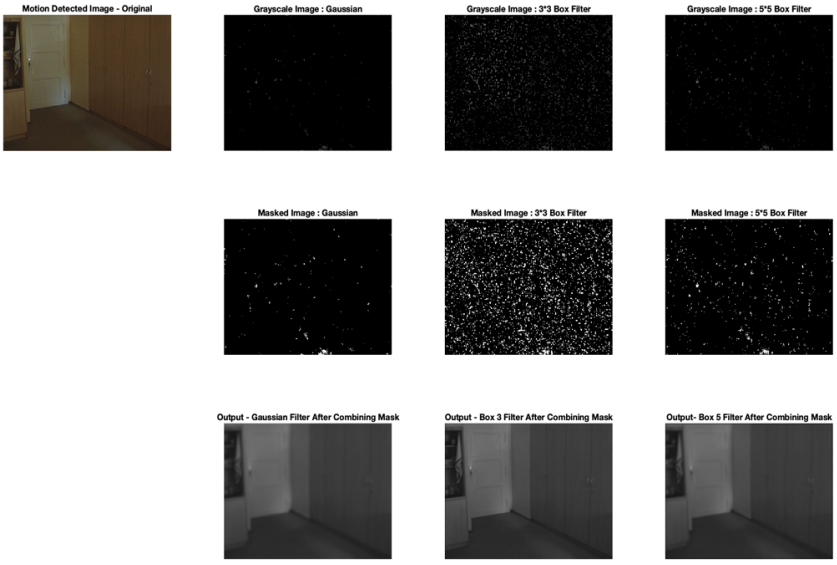


Figure 9

Figure 10

Figure 11

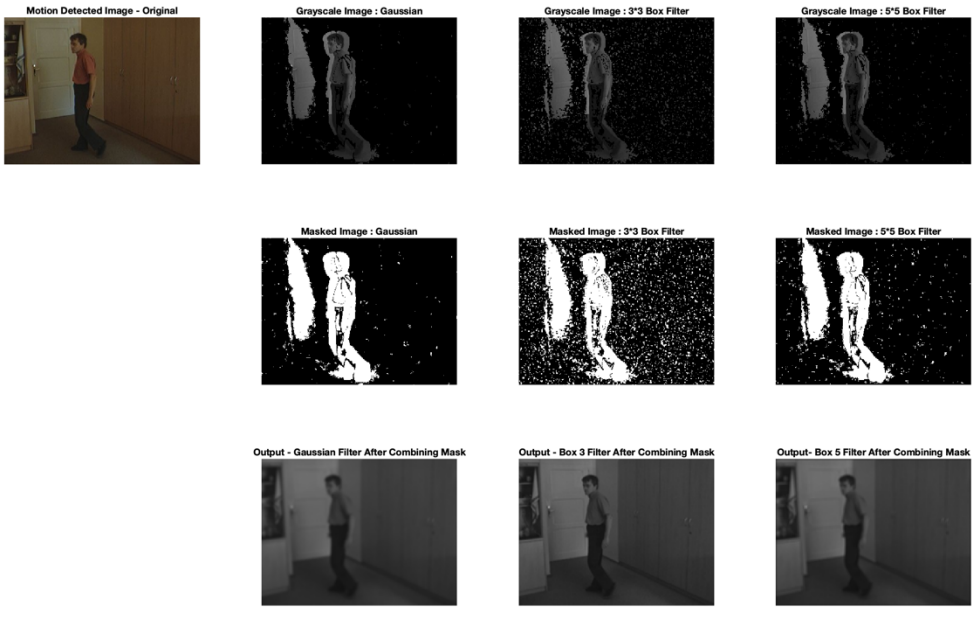


Figure 12

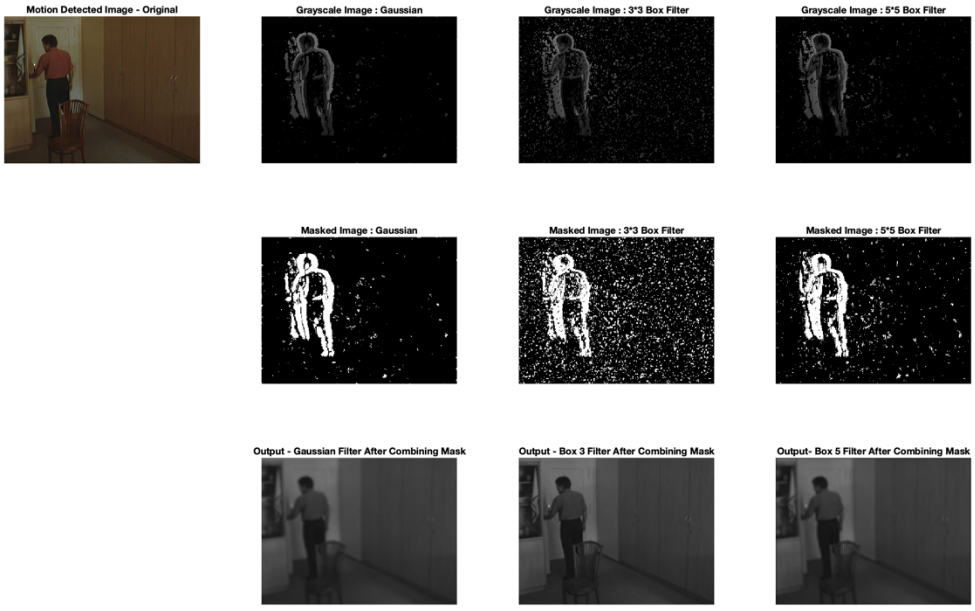


Figure 13

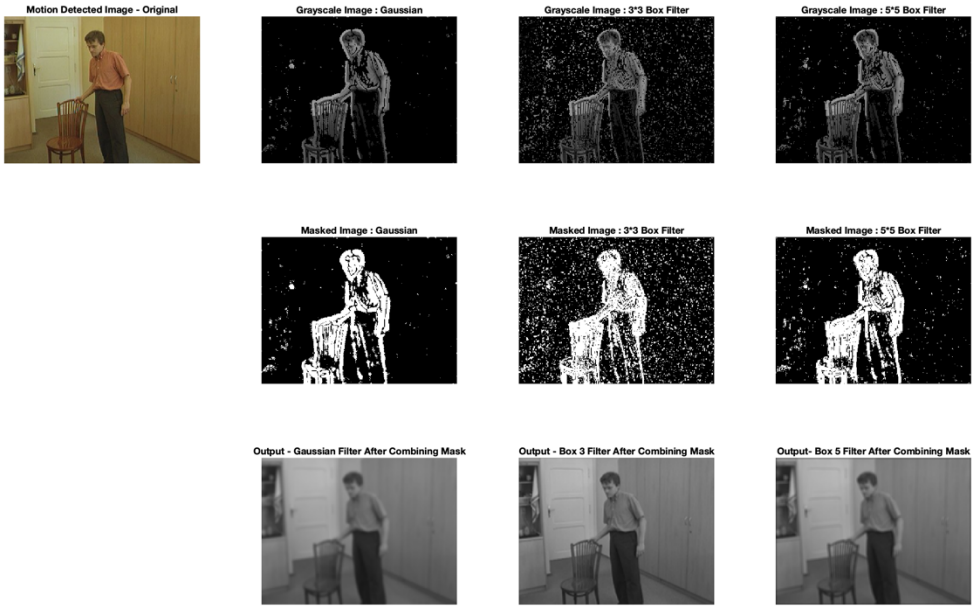
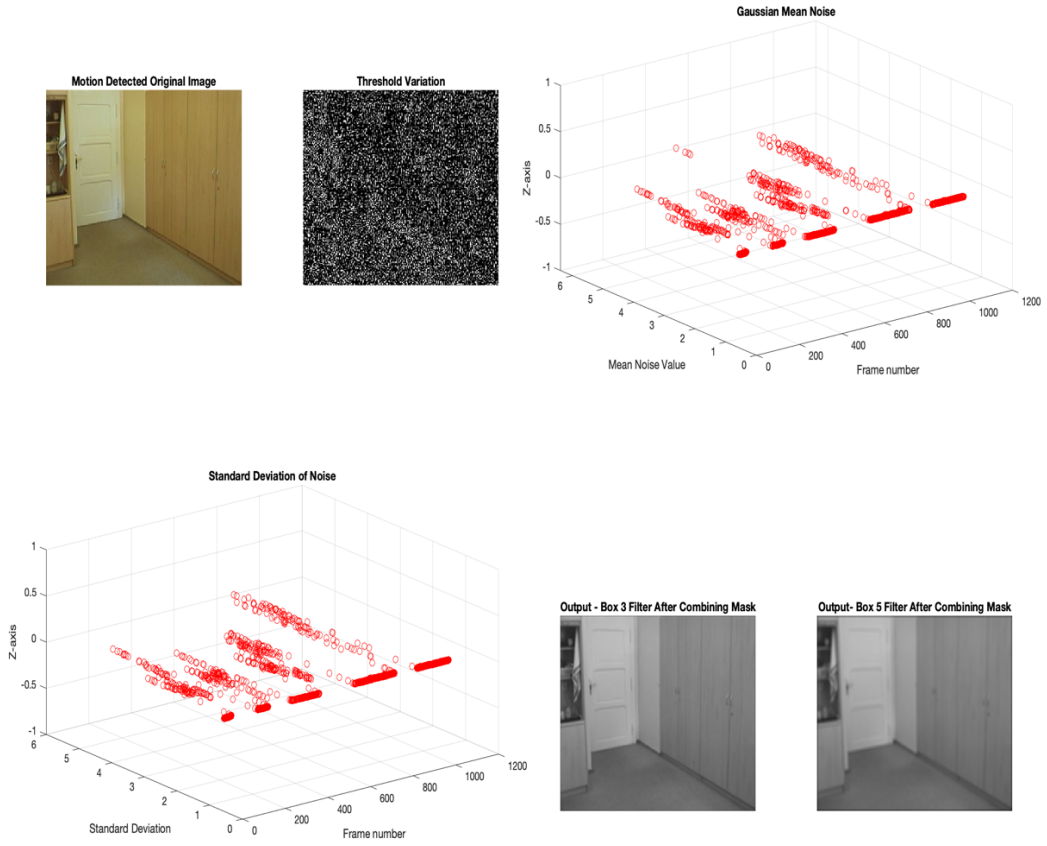


Figure 14



So, in this case, using the threshold value of 2 increases the algorithm's sensitivity, potentially resulting in false detection in the output. It is evident that we can see many black and white masks above, which means using fewer threshold values increase sensitivity and noise, which increases the standard deviation. The motion recorded is scattered in the plot against frames and we can see very high noise and STD as a result. However, the object detection rate is also high using this case due to its high sensitivity.

Threshold value – 6

We varied the threshold value to 6, and below are the recorded plots.

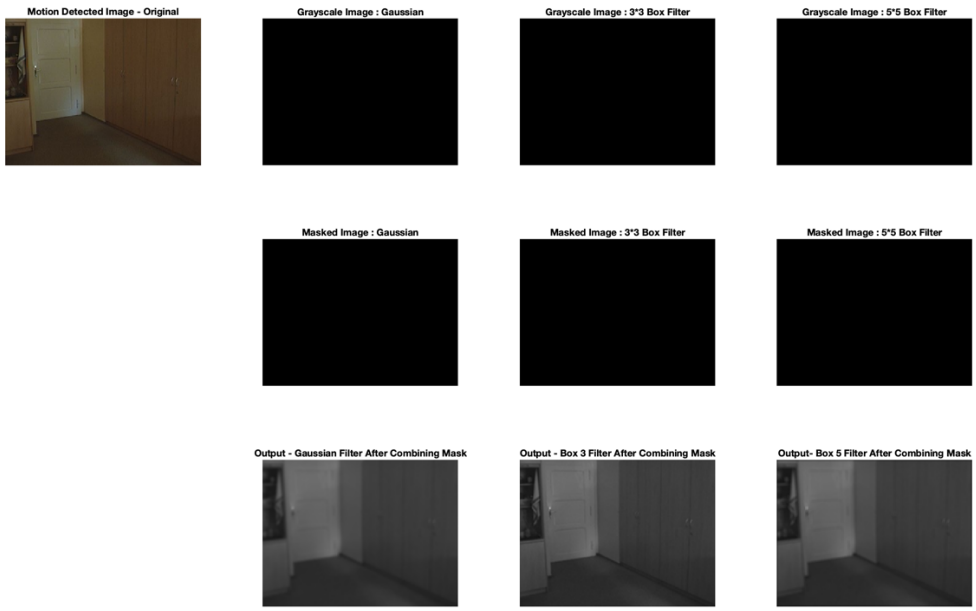


Figure 15

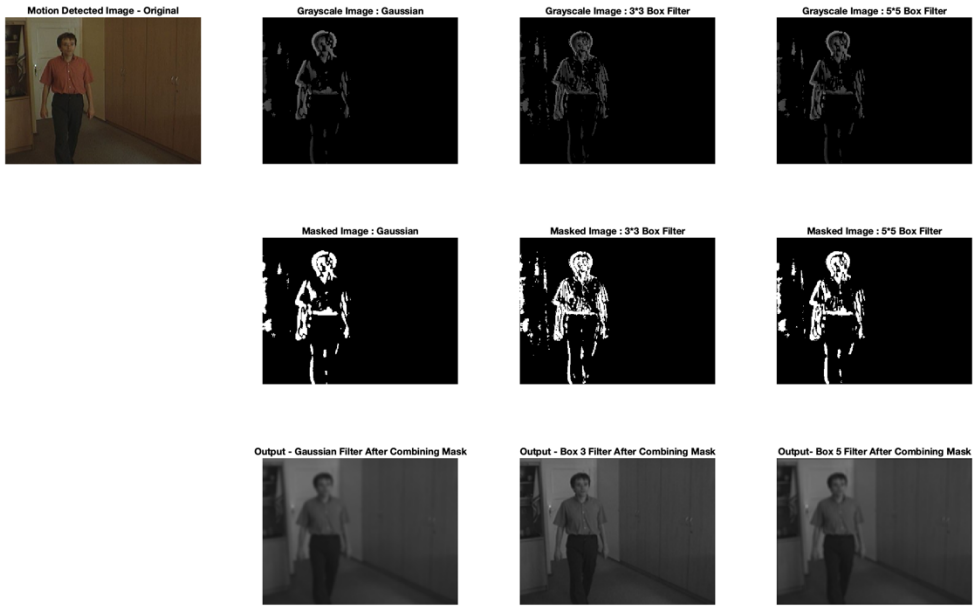


Figure 16

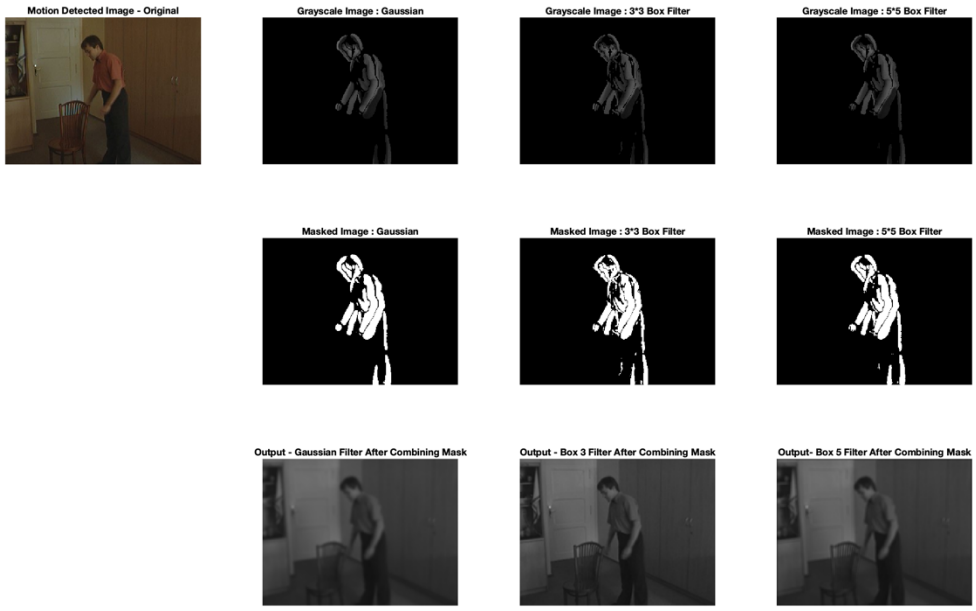


Figure 17

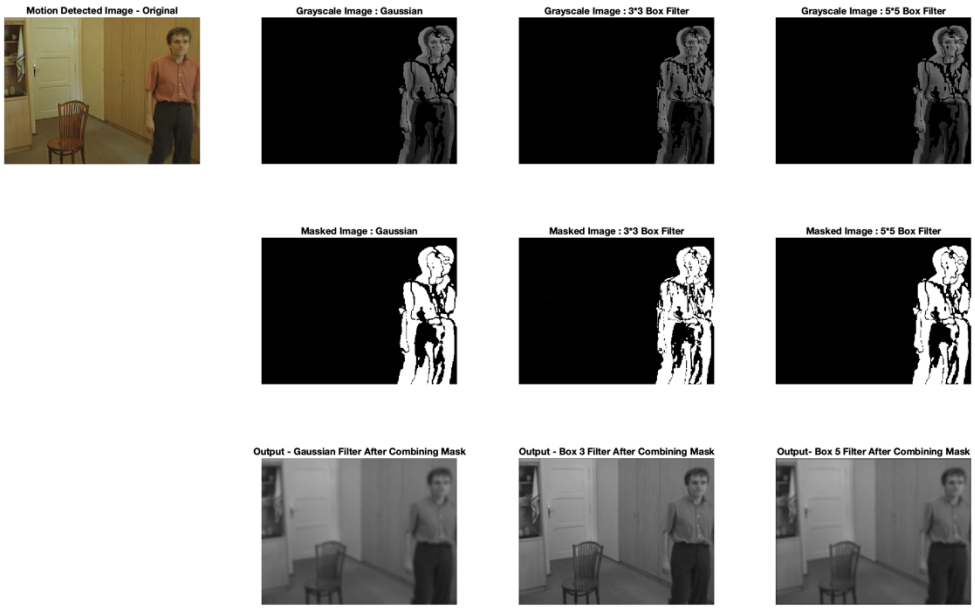


Figure 18

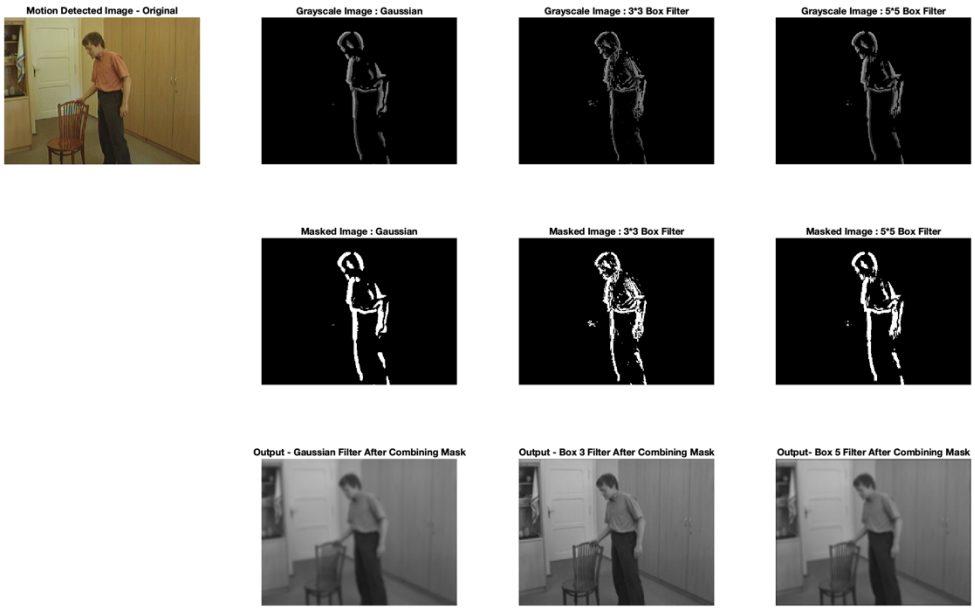


Figure 19

Figure 20

Figure 21

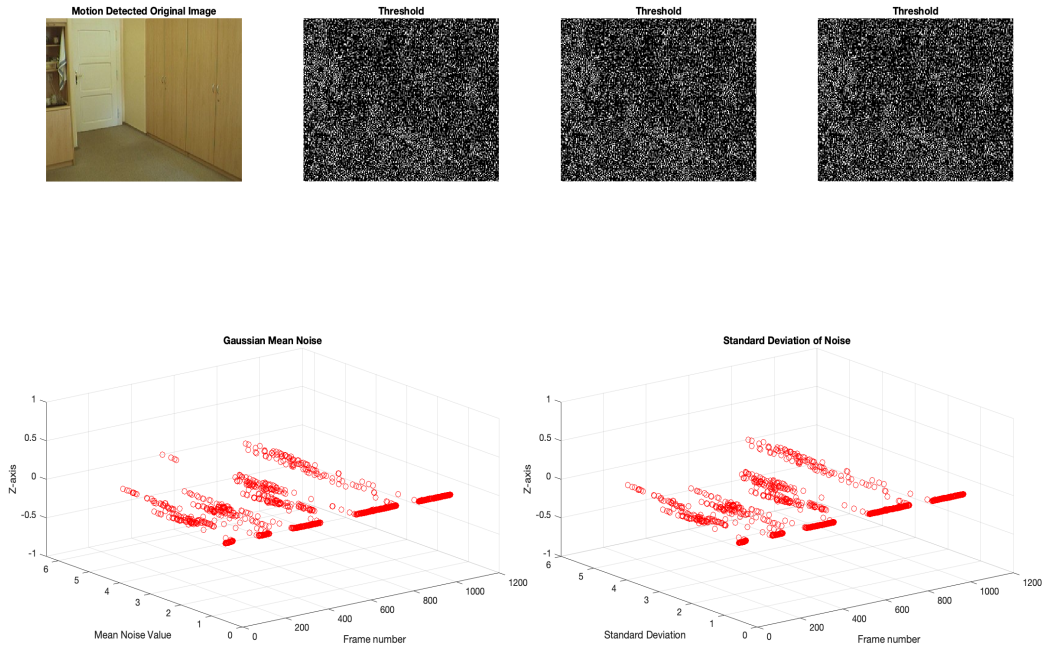


Figure 22

So, in this case, using the threshold value of 6 slightly decreases the algorithm's sensitivity, potentially resulting in good detection in the output. It is evident that we cannot see many black-and-white masks in the above pictures, which means using medium threshold values provides optimal sensitivity, which increases the accuracy of the detection. The motion recorded is scattered in the plot against frames, and we can see nominal noise and STD as a result. Also in this case the object detection rate is also high with optimal threshold and STD.

Case 3 – Threshold value 10

We varied the threshold value to 10, and below are the recorded plots

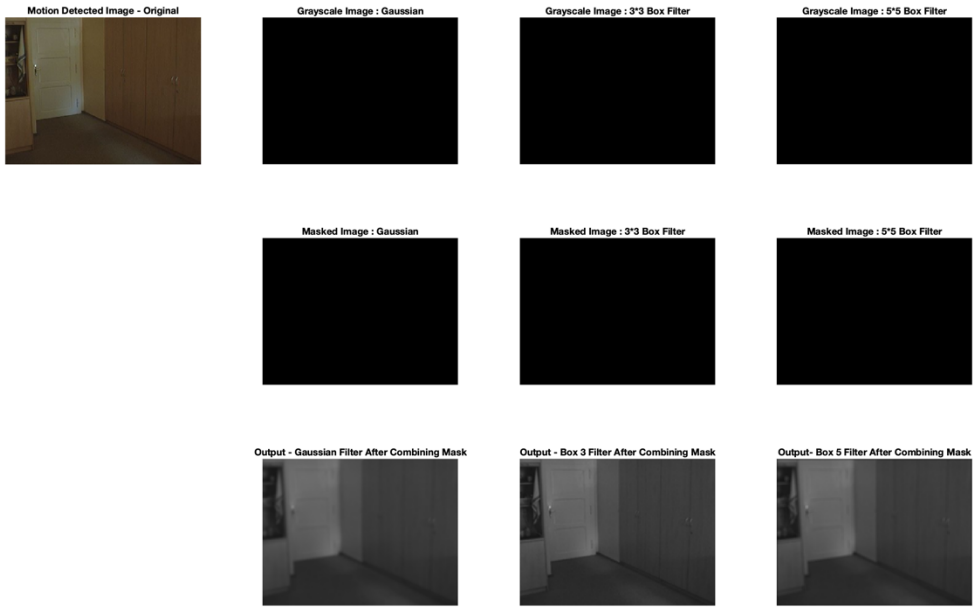


Figure 23

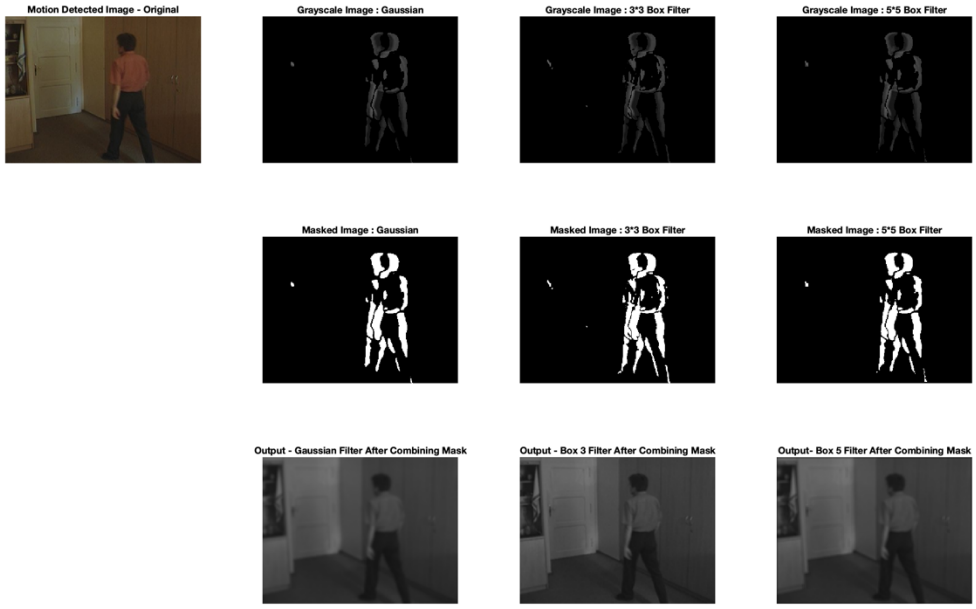


Figure 24

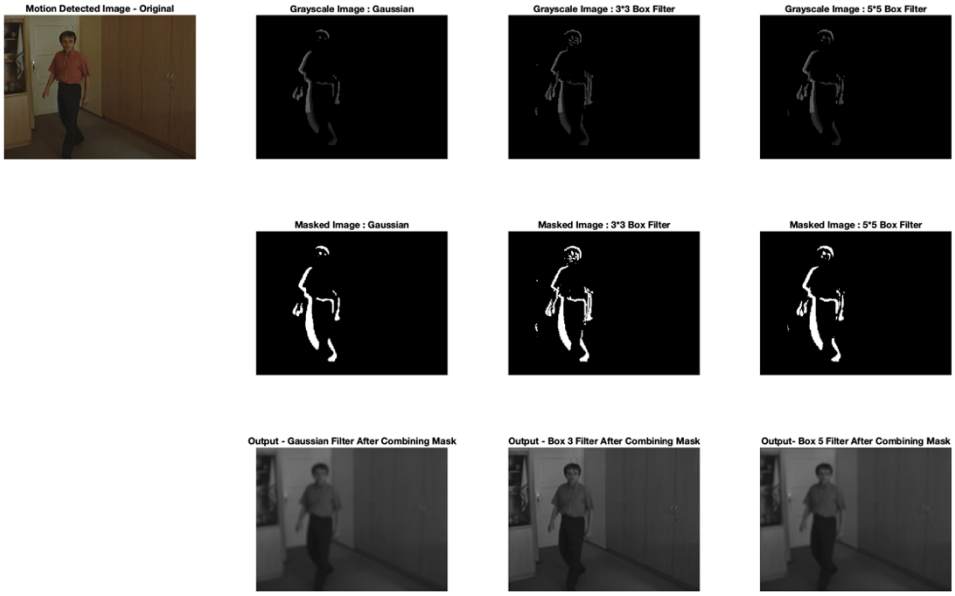


Figure 25

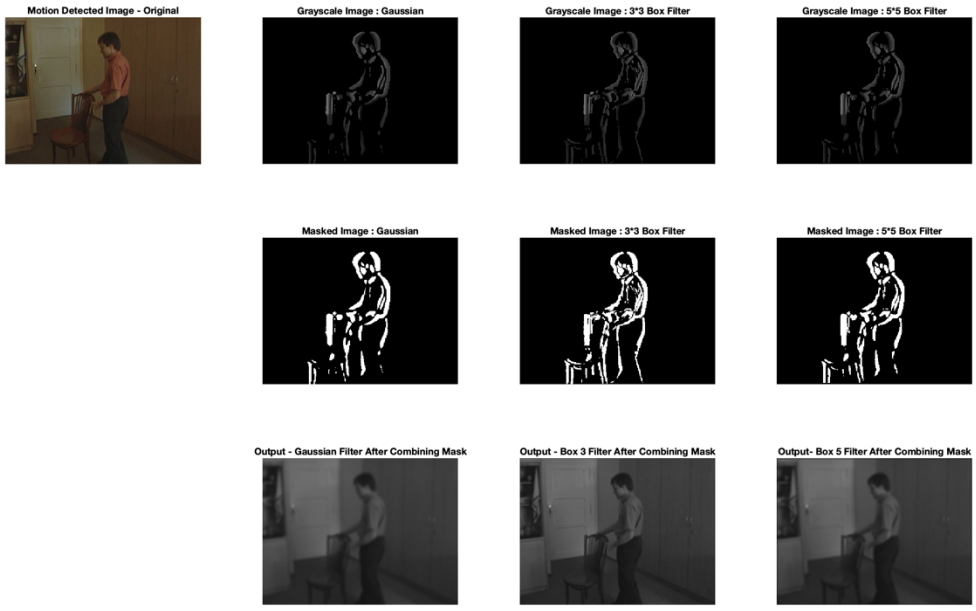


Figure 26

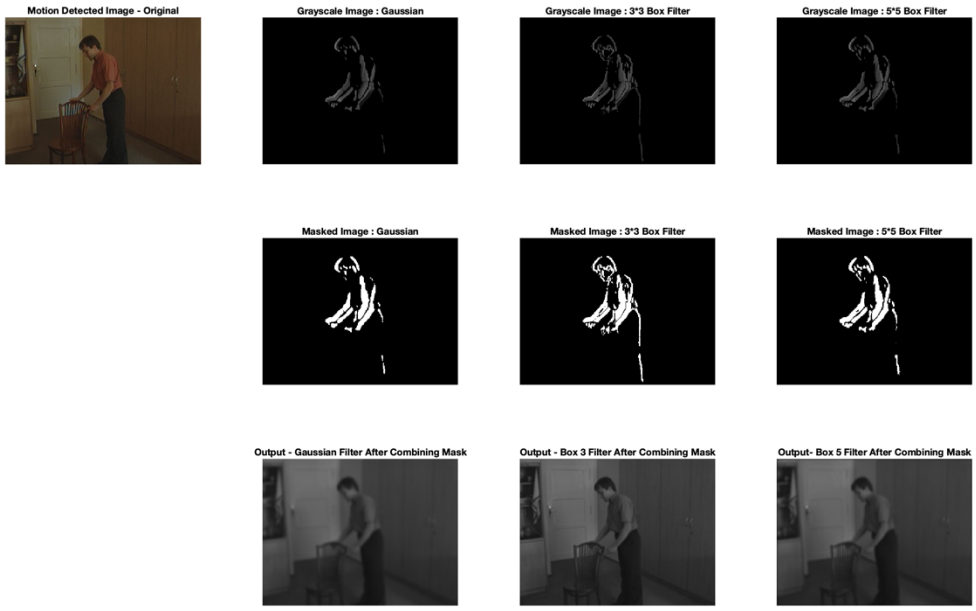


Figure 27

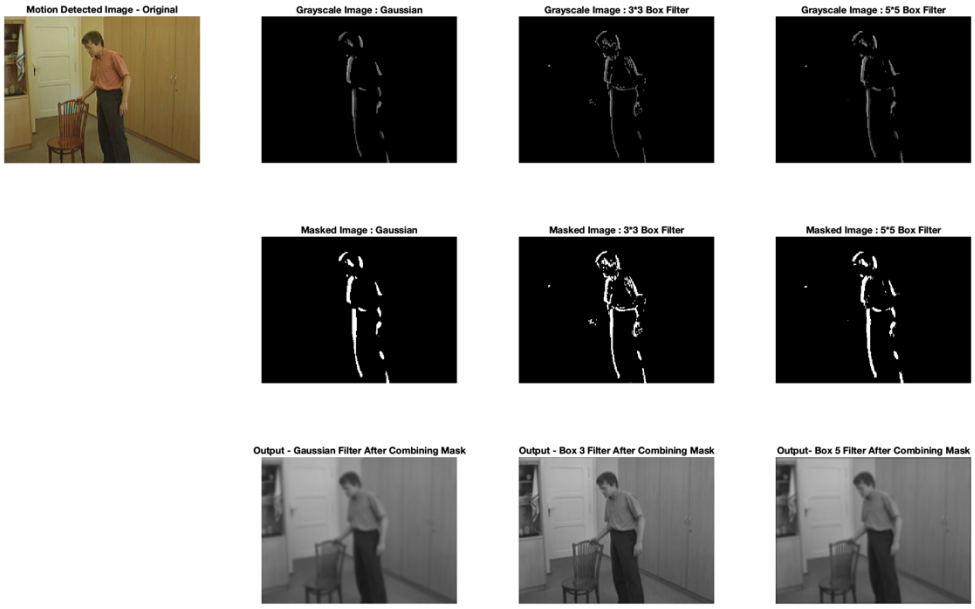


Figure 28

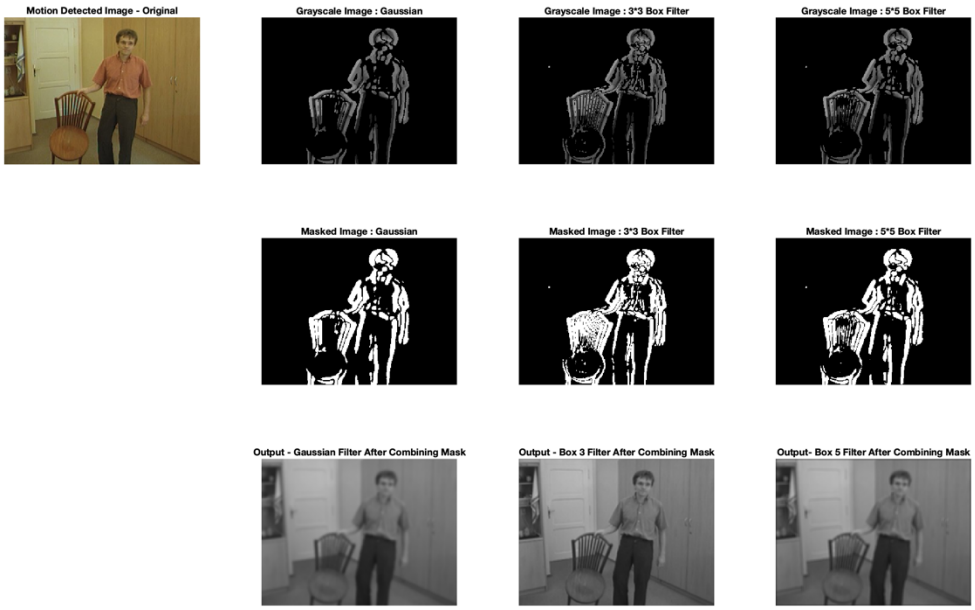


Figure 29

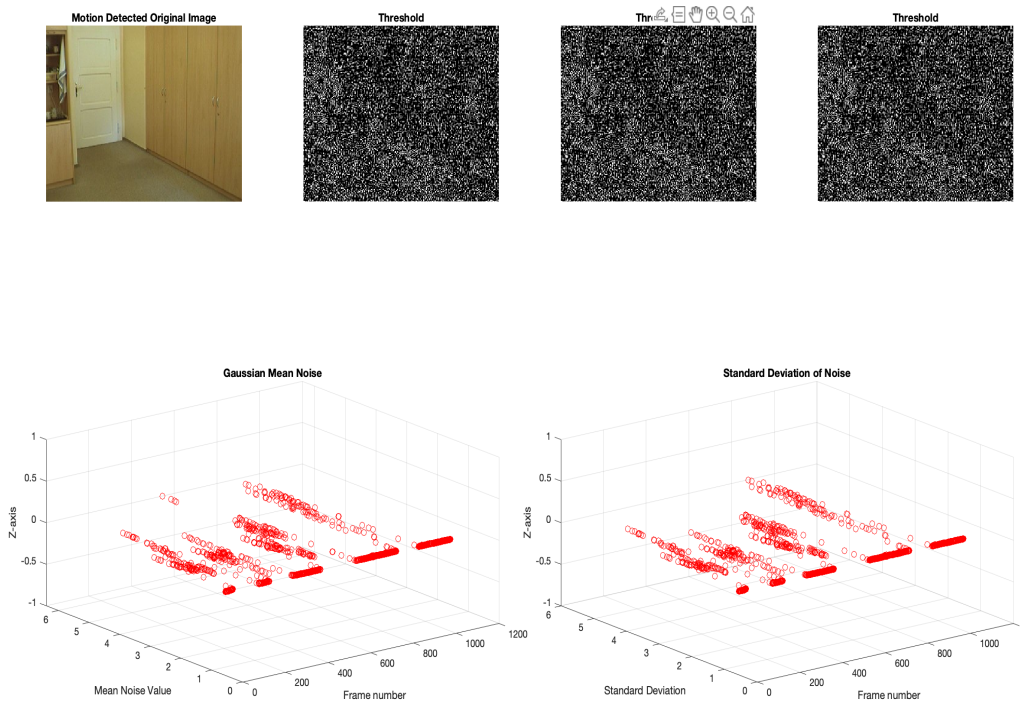


Figure 30

So, in this last case of varying the threshold value to 10, it greatly decreases the algorithm's sensitivity compared to 6 and 2, potentially resulting in better detection in the output but not a great one. From the above pictures, it is evident that in some places, like grabbing the chair, the algorithm misses detecting motion. This means using high threshold values provides less sensitivity, which, in turn, not greatly decreases the accuracy of the detection but to a certain extent. The motion recorded is scattered in the plot against frames, and we can see little noise and STD as a result. Also, in this case, the object detection rate is still better than the very few threshold values. In order to balance noise reduction with maintaining crucial image information, a moderate standard deviation may be more suitable for motion detection in general.

Comparisons and Inference with respect to varying Thresholds

Thresholds are used in motion detection to differentiate between background and foreground movements. While varying Thresholds from 2 to 6 to 10, we observed many differences. While implementing with value 2, the algorithm is very sensitive, which generates enormous false detections. This is due to the presence of much noise. Greater noise increases the deviation, which results in less accuracy.
On the other hand, while experimenting with the value 6, the noise is considerably reduced to a minimal extent. Because of this, the algorithm has optimal sensitivity, which detects the motion with improved accuracy. In this case, the deviation is essentially lesser than in the previous case. Lastly, while experimenting with a threshold value of 10, it almost behaves like the previous one, case 2. Because in this case also, the sensitivity is less, which paved the way to detect the motion better than in case 1. However, due to less sensitivity, it covers almost all major motions and tends to miss some minor motion detections.

The right threshold value to apply relies on the application in question and the features of the processed image. A higher threshold number may be more appropriate if the objective is to remove noise and detect only substantial changes. In contrast, a low threshold value may be more appropriate if the objective is to detect small or subtle changes in the image. On the other hand, when the background is shifting or the foreground motion is minimal, choosing a lower threshold number can be helpful. However, if the threshold value is too low, non-motion areas could be mistakenly identified as foreground motion, leading to false positives. Hence in this case the threshold values between 6 to 10 will be optimal.

Conclusion

Using a low threshold, in general, can increase the sensitivity of an object detection or segmentation algorithm, potentially resulting in more false positives or noise in the output. This is because a low threshold value will classify more pixels as foreground or object, even if they are in the background or noise. When identifying only multiple motions against a background that is mainly stationary, using a more significant threshold value can be helpful. But if the threshold value is too high, it can miss some motion types or cause delayed detection. However, whether a low threshold produces more noise depends on the algorithm and the characteristics of the image being processed. Some algorithms are built to deal with noise and other false positives, while others are not. So, we can infer that better motion detection relies not entirely on the threshold but also on various aspects such as noise, standard deviation, image characteristics, background images, etc. Hence, it is critical to select an appropriate threshold value based on the specific application and image characteristics. It frequently requires trial and adjusting to get the best outcomes in a particular circumstance.

APPENDIX : CODE

```
% EECE5639 - COMPUTER VISION
% Code for Motion Detector - Project 1

parameters = {
    'White', 255;
    'Black', 0;
    'Box3_filter', '3x3Box';
    'Box5_filter', '5x5Box';
    'Gauss_filter', 'gauss';
    'Temporal_gradient', 1;
    'Threshold_Value', 2;
    'Gauss_Stand_dev', 2
};

for i = 1:size(parameters, 1)
    eval(sprintf('global %s;', parameters{i, 1}));
    eval(sprintf('%s = %d;', parameters{i, 1},
parameters{i, 2}));
end

function motion_detector()

% Specifying the location of the directory and checking
if the file exists
    input_directory = '/Users/dhanush/Downloads/Office
2/';
if exist(input_directory, 'dir') == 0
    error('The following folder does not exist: Please
check\n%s', input_directory);
end

% Implementation of processing_pipeline function for
video

    video = imageDatastore(input_directory)
    processing_pipeline(video)
end

function processing_pipeline(video)
% Starting to process the pipeline
```

```
    frame_number = 2;
    while frame_number < length(video.Files)
       previous_frame = readimage(video, frame_number -
1) * Temporal_gradient;
       next_frame = readimage(video, frame_number + 1) *
Temporal_gradient;
       frame.color = readimage(video, frame_number);

       filter_params = {GAUSS, BOX3, BOX5};
       filter_names = {'gauss', 'box3', 'box5'};

for i = 1:numel(filter_params)
    [filter, mask, movement] =
processing_pipeline_helper(frame.color, previous_frame,
next_frame, filter_params{i});
    frame.([filter_names{i} '_filter']) = filter;
    frame.([filter_names{i} '_mask']) = mask;
    frame.([filter_names{i} '_movement']) = movement;
end

    display_results(frame);

    frame_number = frame_number + 1;
    end

% Implementation of Gaussian 1D filter 1/16 * [1 4 6 4 1]
over each pixel with respect to time.

frame_number = 3;
      while frame_number <= length(video.Files) - 5
      kernel_frame = getGaussCube(video,
size(readimage(video, 1)));
      temporal_frame = getTemporalCube(video,
frame_number);
      differentialImage = convn(double(temporal_frame),
kernel_frame, 'valid');


% Threshold values - defining the threshold value and
creates the maskd
        mask = delta_threshold(differentialImage);
```

```
        color = readimage(video, frame_number);
        movement = uint8(double(rgb2gray(color)) .*
double(mask/WHITE));
% Image
        %Plotting Original Image – Motion Detected
        subplot(2,4,1);
        imshow(color);
        title('Motion Detected Original Image')

        %Plotting Threshold
        subplot(2,4,2);
        imshow(differentialImage);
        title('Threshold')

        %Plotting Threshold
        subplot(2,4,3);
        imshow(differentialImage);
        title('Threshold')

        %Plotting Threshold
        subplot(2,4,4);
        imshow(differentialImage);
        title('Threshold')

        %Plotting Gaussian Mean Noise
        subplot(2, 4, [5, 6]);
        scatter3(k, (sqrt(mean2(differentialImage .*
differentialImage))),zeros(size(k)), 'r');
        title('Gaussian Mean Noise')
        xlabel('Frame number')
        ylabel('Mean Noise Value')
        zlabel('Z-axis')
        hold on;

        %Plotting Standard Deviation of Noise
        subplot(2, 4, [7, 8]);
        scatter3(k,
std2(differentialImage),zeros(size(k)), 'r');
        title('Standard Deviation of Noise')
        xlabel('Frame number')
        ylabel('Standard Deviation')
```

```
        zlabel('Z-axis')
        hold on;
        drawnow;


      frame_number=frame_number+1;
      end

end

function gauss_frame = getGaussCube(video, imageSize)
    gauss = [1 4 6 4 1] * (1/16);
    differential = [-2 0 2];
    kernel = conv2(differential, gauss, 'same');
    differentialImage = ones(imageSize(1), imageSize(2));
    gauss_frame = repmat(kernel, imageSize(1),
imageSize(2), 1);
end

function slice = getTemporalCube(video, k)
    img1 = rgb2gray(readimage(video, k-1));
    img2 = rgb2gray(readimage(video, k));
    img3 = rgb2gray(readimage(video, k+1));
    slice = cat(3, img1, img2, img3);
end

% Define filter functions and corresponding labels
filters = {@(frames) imfilter(frames, (1/9) * ones(3),
'replicate'), '3x3Box'; ...
           @(frames) imfilter(frames, (1/25) * ones(5),
'replicate'), '5x5Box'; ...
           @(frames) imgaussfilt(frames,
Gauss_Stand_dev), 'gauss'};

% Define the main function
function [filter, mask, movement] =
processing_pipeline_helper(color, previous_frame,
next_frame, filter_selector)
    [previous_frame, next_frame] =
apply_spatial_filter(previous_frame, next_frame,
filter_selector);
```

```
    filter = rgb2gray(next_frame);
    delta = abs(double(rgb2gray(next_frame)) -
double(rgb2gray(previous_frame)));

    mask = delta_threshold(delta);
    movement = uint8(double(rgb2gray(color)) .*
(mask/WHITE));

    function [previous_frame, next_frame] =
apply_spatial_filter(previous_frame, next_frame,
selector)
        idx = find(strcmp(selector, filters(:, 2)));
        previous_frame = filters{idx, 1}(previous_frame);
        next_frame = filters{idx, 1}(next_frame);
    end
end


function mask = delta_threshold(frame)

    frame(abs(frame) >= Threshold_Value) = WHITE;
    frame(abs(frame) < Threshold_Value) = 0;

    mask = frame;
end

function display_results(frame)

    % Image
    subplot(3,4,1);
    imshow(frame.color);
    title('Motion Detected Image - Original')


    % Plot gray scale image of Gaussian
    subplot(3,4,2);
    imshow(frame.gauss_movement);  % Display mask.
    title('Grayscale Image : Gaussian')

    % Plot gray scale image of Box3
```

```
    subplot(3,4,3);
    imshow(frame.box3_movement);  % Display mask.
    title('Grayscale Image : 3*3 Box Filter')

    % Plot gray scale image of Box5
    subplot(3,4,4);
    imshow(frame.box5_movement);  % Display mask.
    title('Grayscale Image : 5*5 Box Filter')


    % Plot Gaussian Mask
    subplot(3,4,6);
    imshow(frame.gauss_mask); % Multiply by WHITE value
to get a binary image
    title('Masked Image : Gaussian')

     %Plot 3*3 Mask
    subplot(3,4,7);
    imshow(frame.box3_mask); % Multiply by WHITE value to
get a binary image
    title('Masked Image : 3*3 Box Filter')

    % Plot 5*5 Mask
    subplot(3,4,8);
    imshow(frame.box5_mask); % Multiply by WHITE value to
get a binary image
    title('Masked Image : 5*5 Box Filter')

    % Plot Output of Gaussian Filter
    subplot(3,4,10);
    imshow(frame.gauss_filter);
    title('Output - Gaussian Filter After Combining
Mask')


    % Plot Output of 3*3 Filter
    subplot(3,4,11);
    imshow(frame.box3_filter);
    title('Output - Box 3 Filter After Combining Mask')

    % Plot Output of 5*5 Filter
```

```
    subplot(3,4,12);
    imshow(frame.box5_filter);
    title('Output- Box 5 Filter After Combining Mask')
    drawnow;
end
```

Citation : Referred Professor Octavia's Lectures, Slides, Various MATLAB Tutorials, Functions from Internet, Resources for understanding SD and Masks