

Laporan Tugas 2

Interpretasi dan Pengolahan Citra

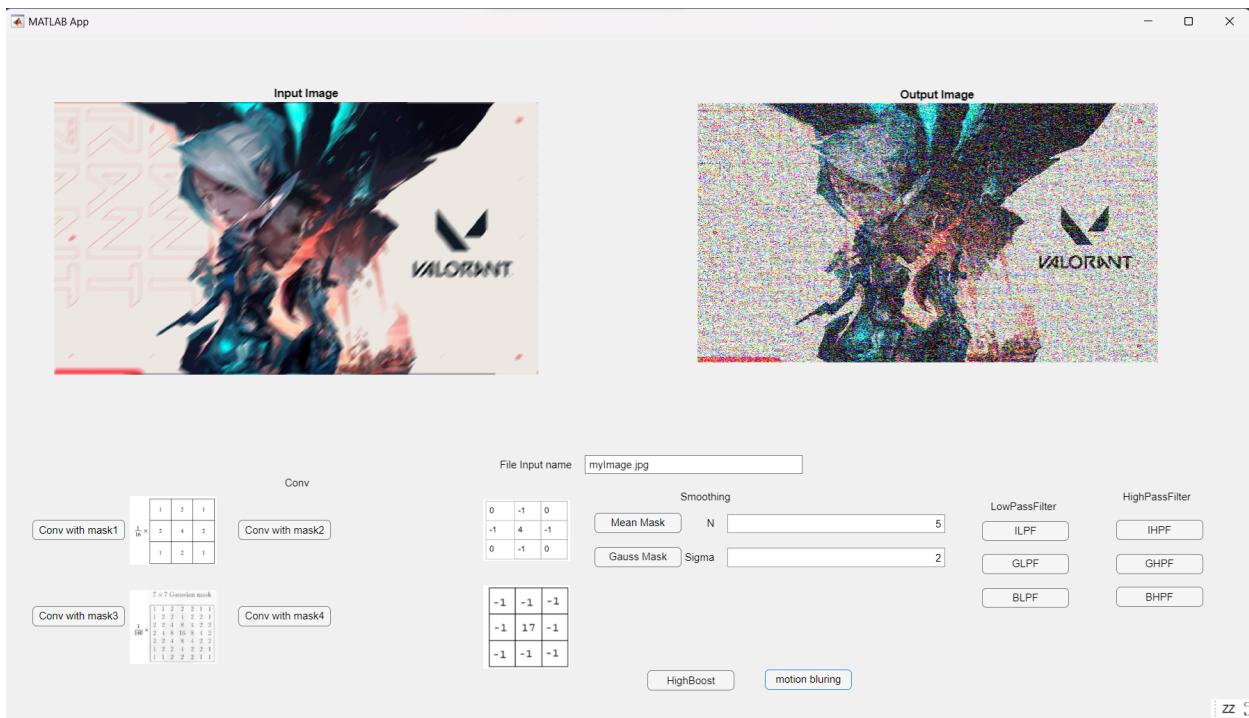


Disusun oleh:
13519008 Ronggur Mahendra

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2023

1. Screenshot GUI



2. Program

a. Konvolusi

i. Kode Program

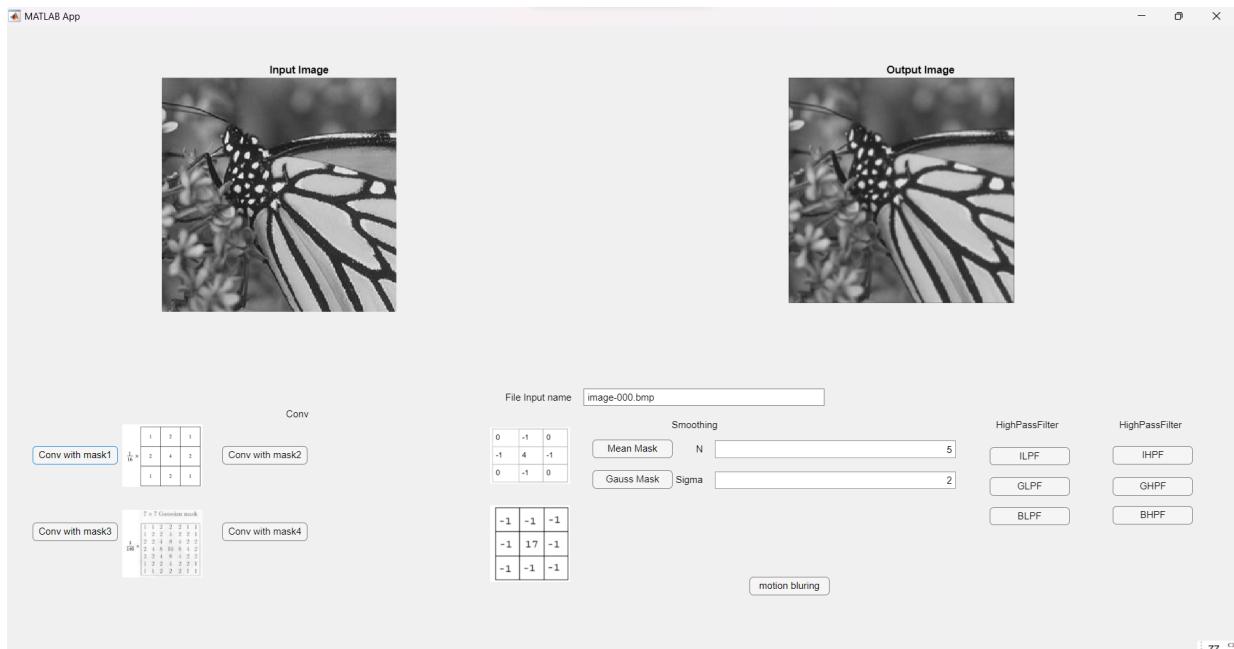
```
% Button pushed function: ConvWithMask2Button
function ConvWithMask2ButtonPushed(app, event)
    filename = app.FileInputNameEditField.Value;
    img_input = imread(filename);
    imshow(img_input, 'Parent', app.UIAxes);
    % mask1 = [1/16 2/16 1/16 ; 2/16 4/16 2/16 ; 1/16 2/16 1/16];
    mask1 = [1 2 1 ; 2 4 2 ; 1 2 1]/16;
    mask2 = [0 -1 0 ; -1 4 -1 ; 0 -1 0];
    mask3 = [1 1 2 2 2 1 1 ; 1 2 2 4 2 2 1 ; 2 2 4 8 4 2 2 ; 2 4 8 16 8 4 2 ; 2 2 4 8 4 2 2 ; 1 2 2 4 2 2 1 ; 1 1 2 2 2 1 1] / 140 ;
    mask4 = [-1 -1 -1 ; -1 17 -1 ; -1 -1 -1];
    img_result = uint8(my_rgb_conv(double(img_input), mask2));
    imshow(img_result, 'Parent', app.UIAxes_2);
end
```

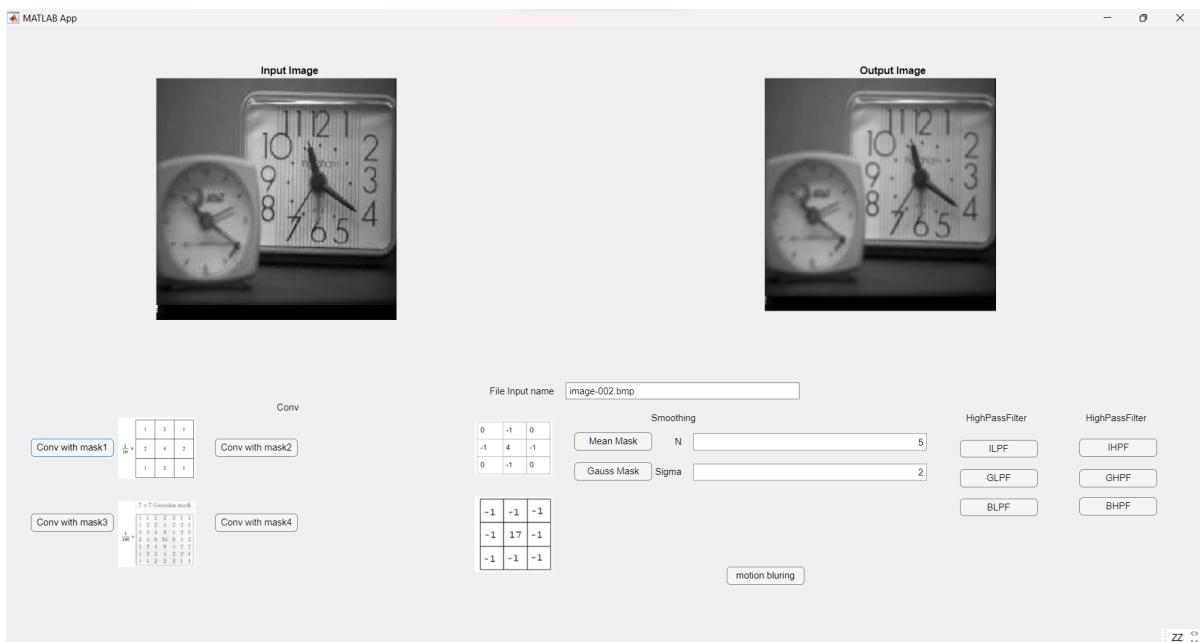
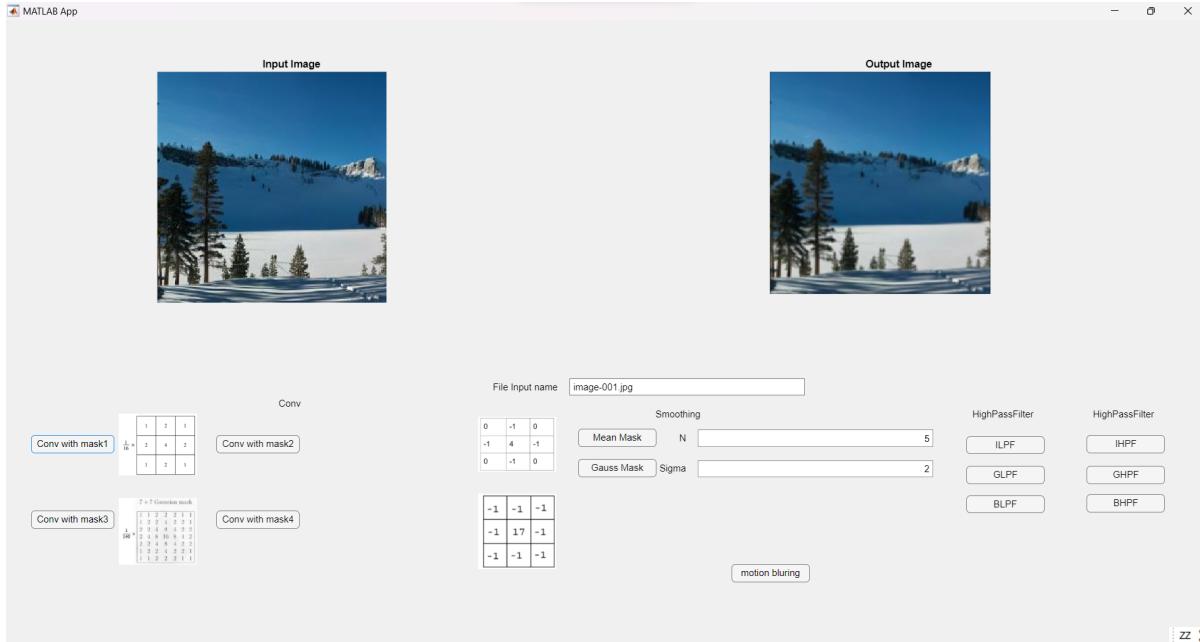
Cat : dengan parameter masukan disesuaikan dengan tombol pada GUI

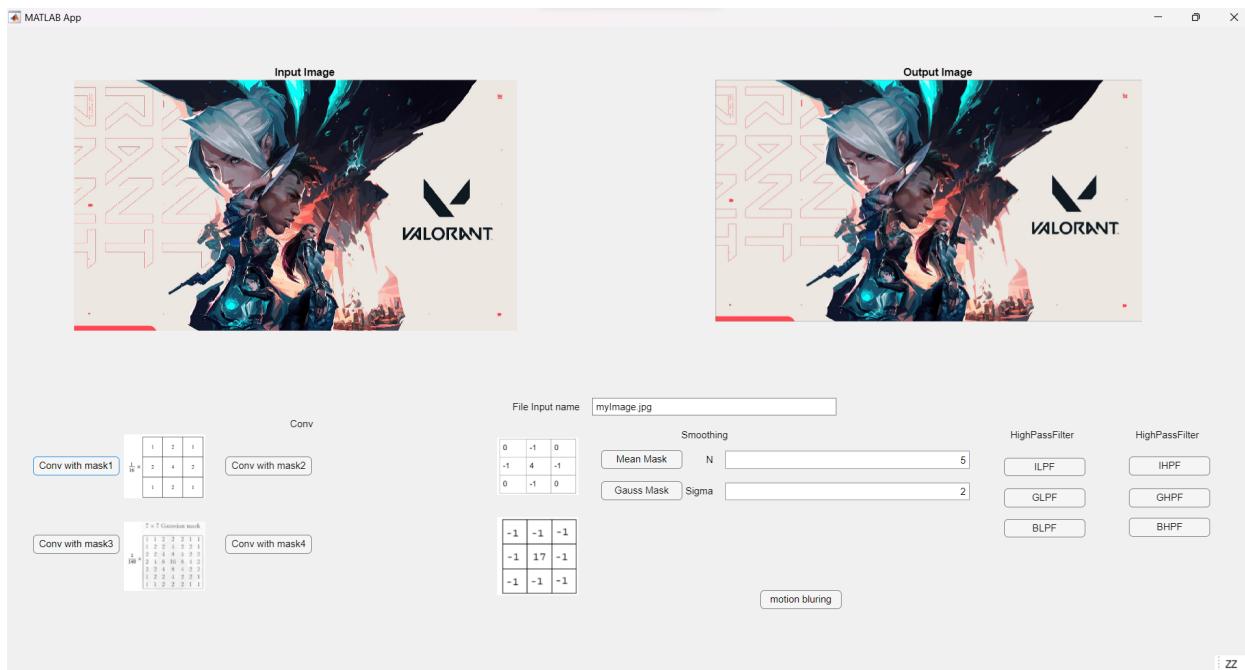
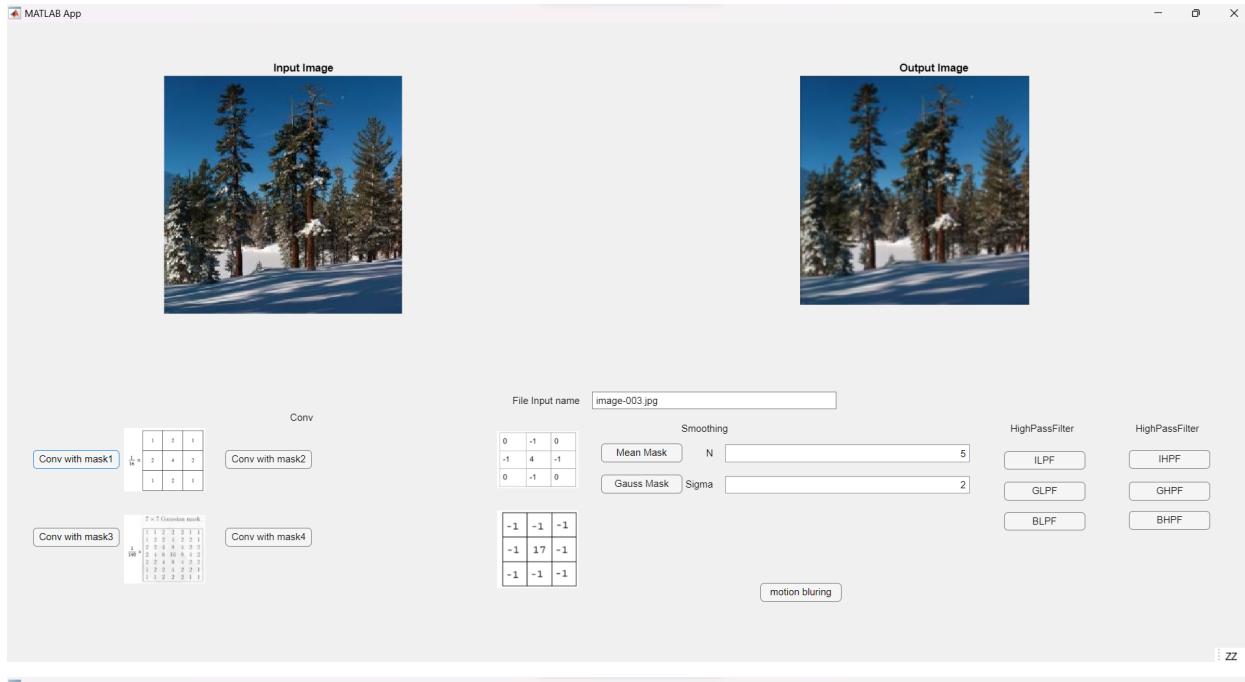
ii. Hasil Eksekusi

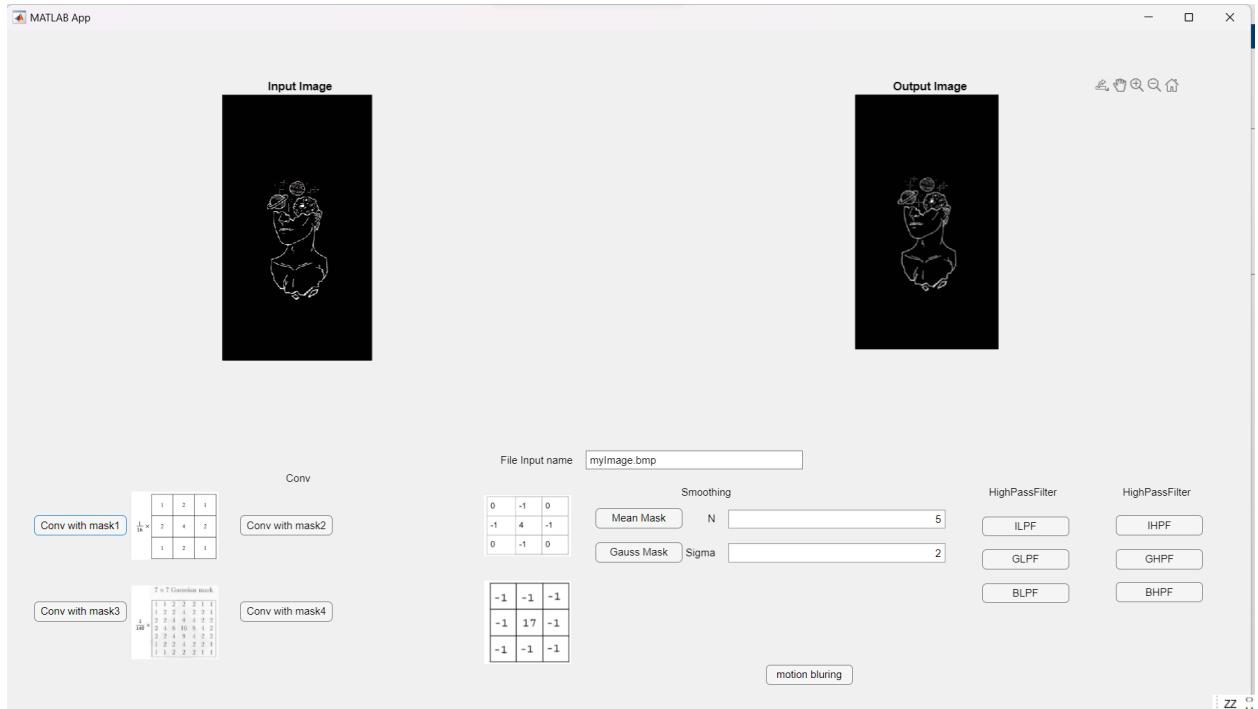
1. Mask 1

$$\frac{1}{16} \times \begin{matrix} & 1 & 2 & 1 \\ & 2 & 4 & 2 \\ 1 & 2 & 1 & \end{matrix}$$



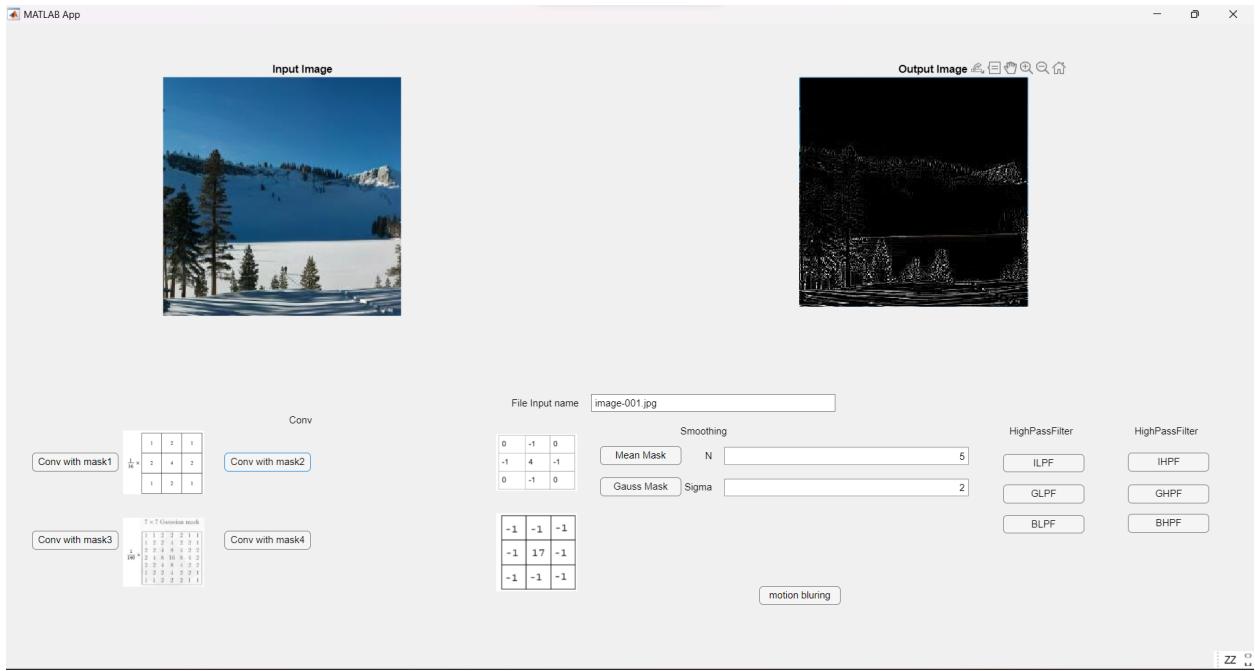
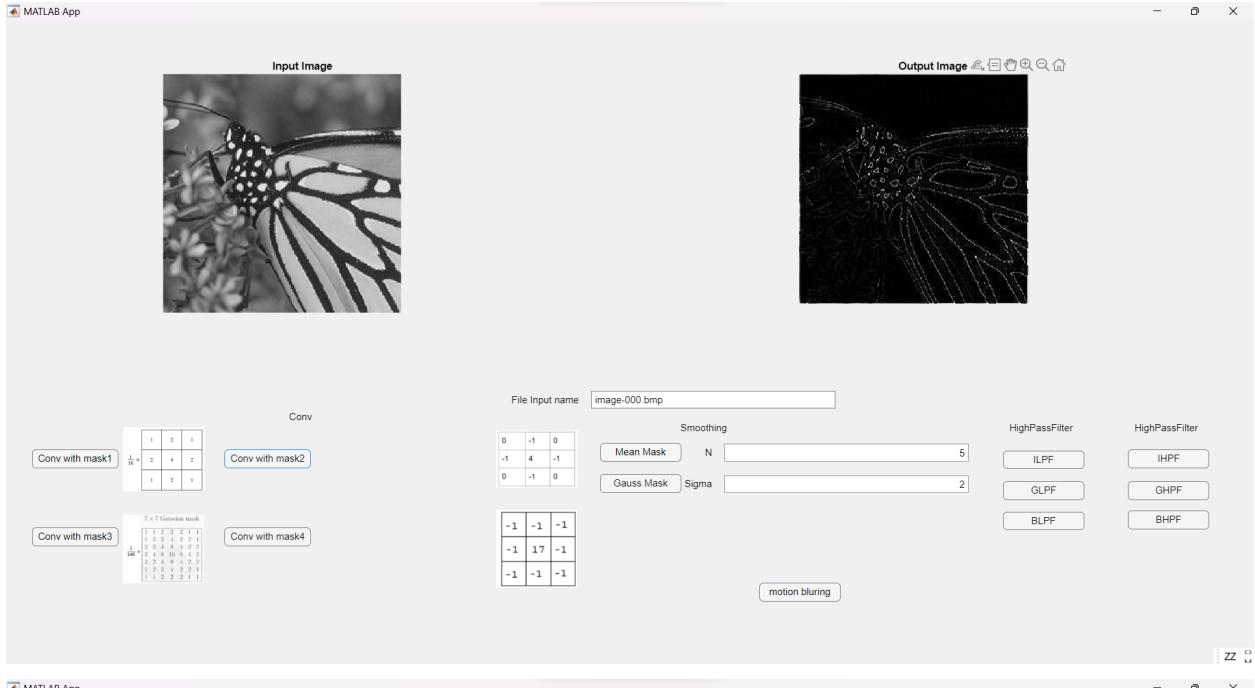


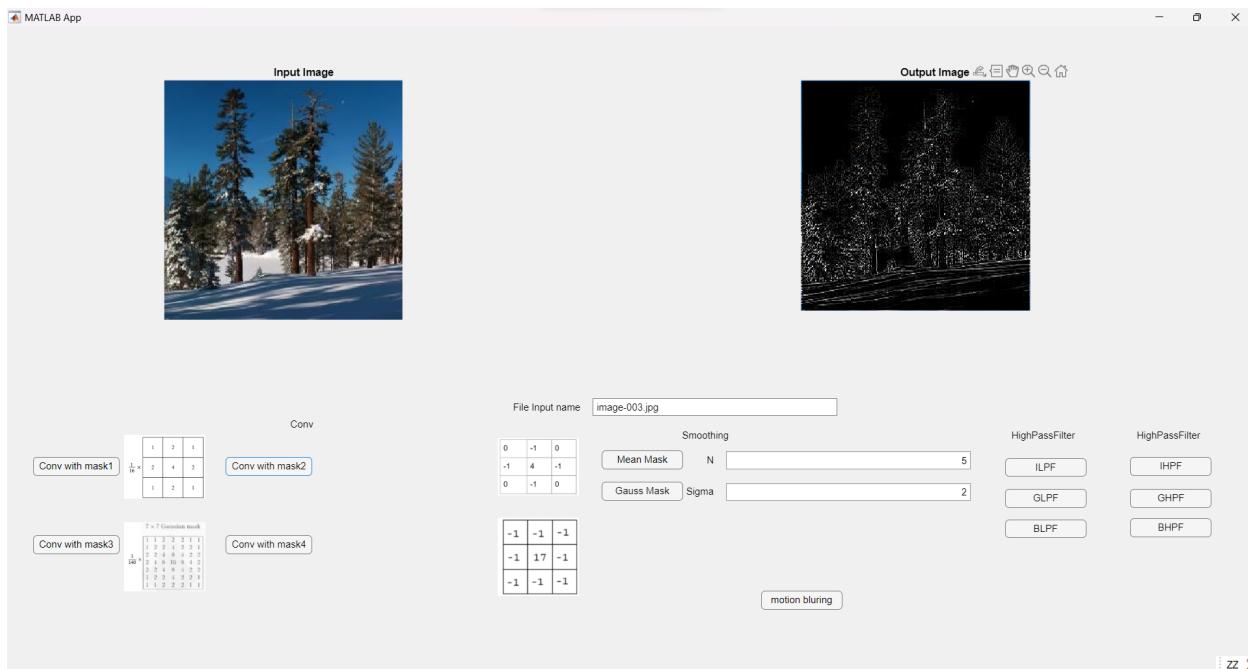
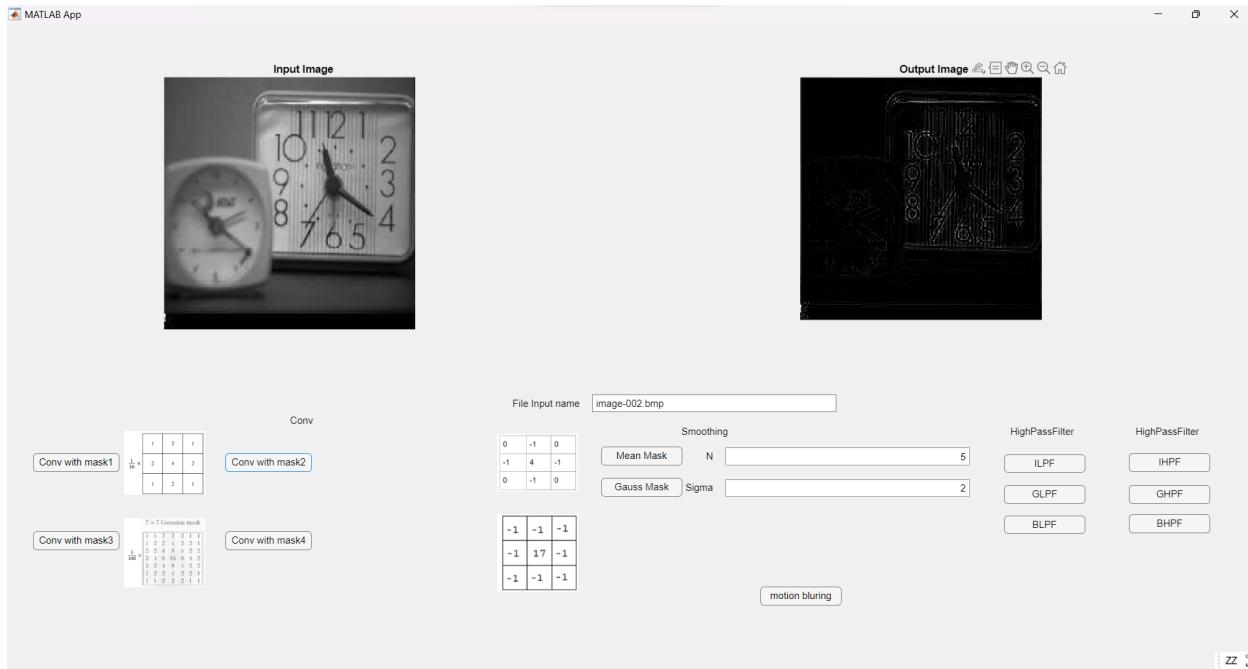


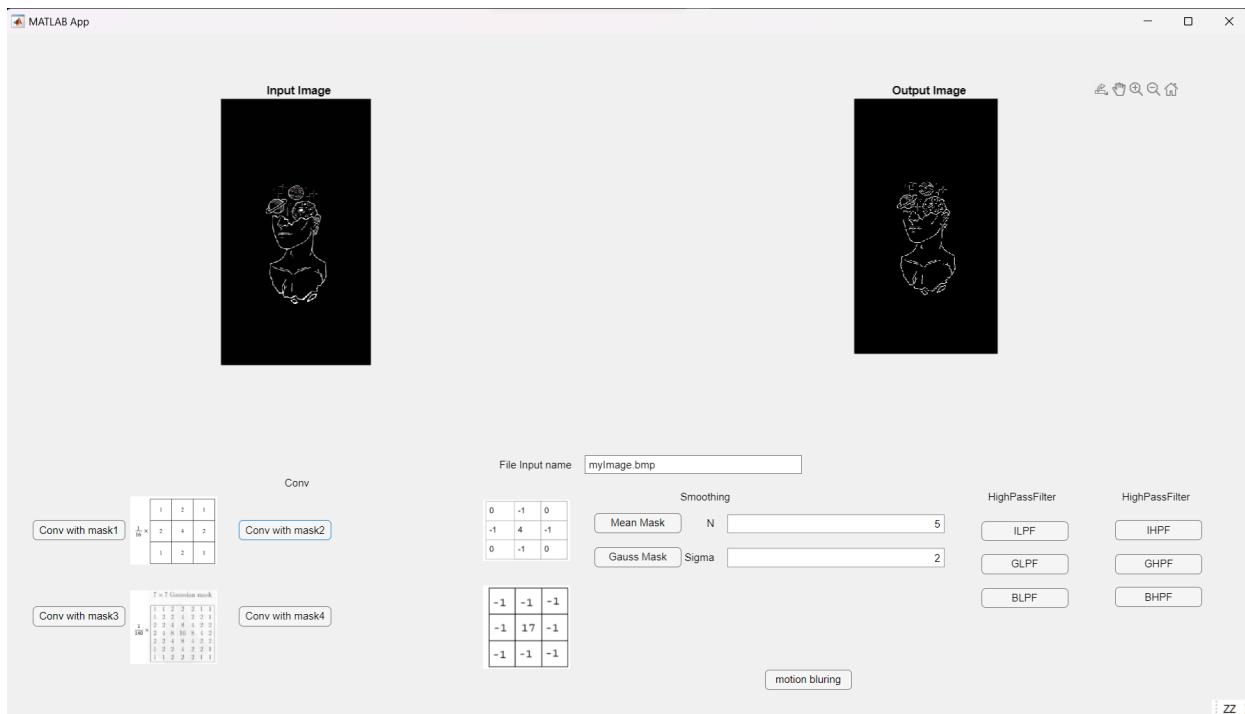
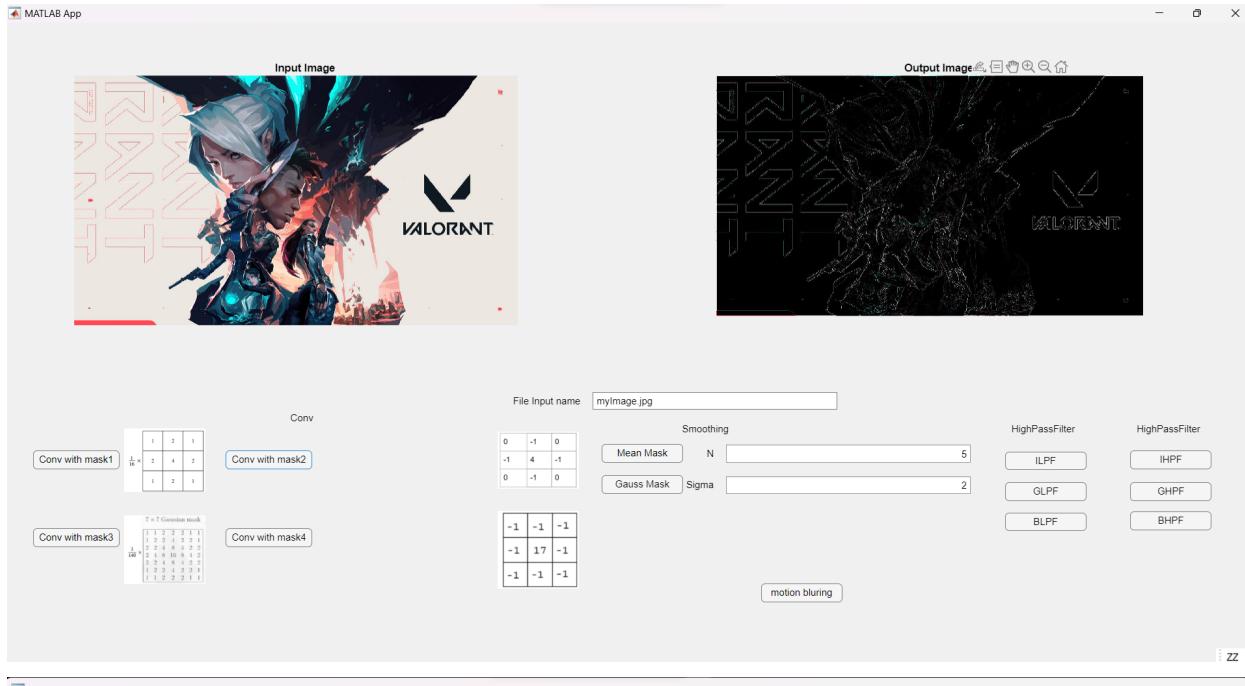


2. Mask 2

0	-1	0
-1	4	-1
0	-1	0



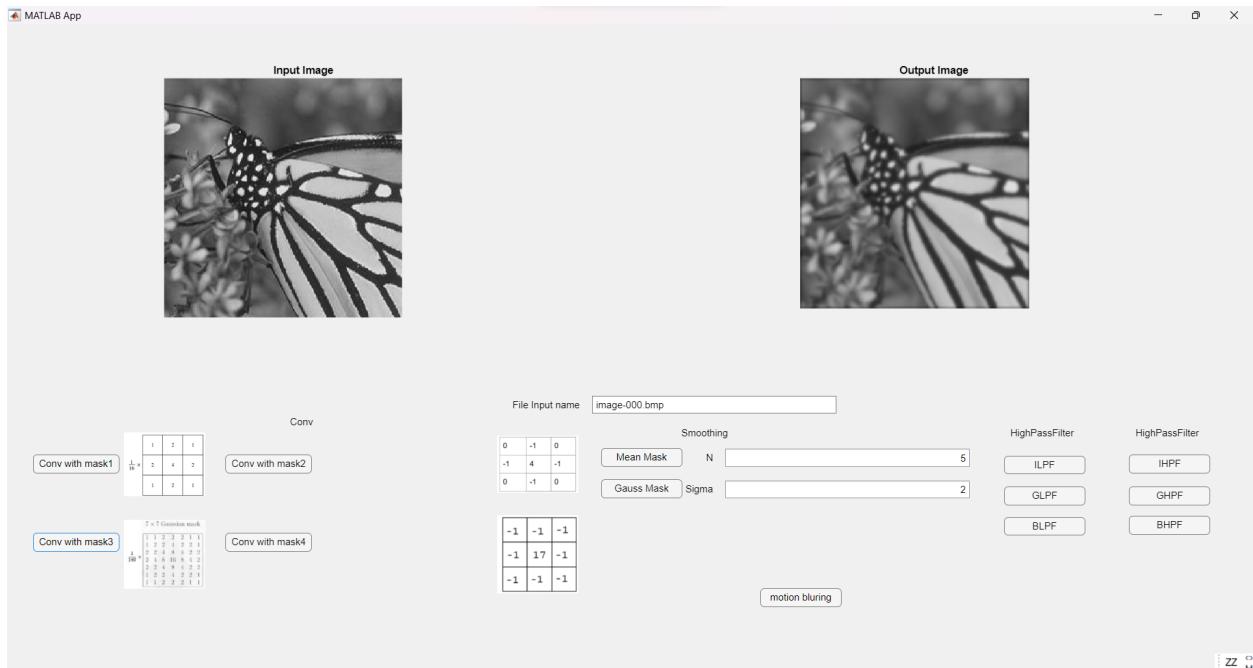


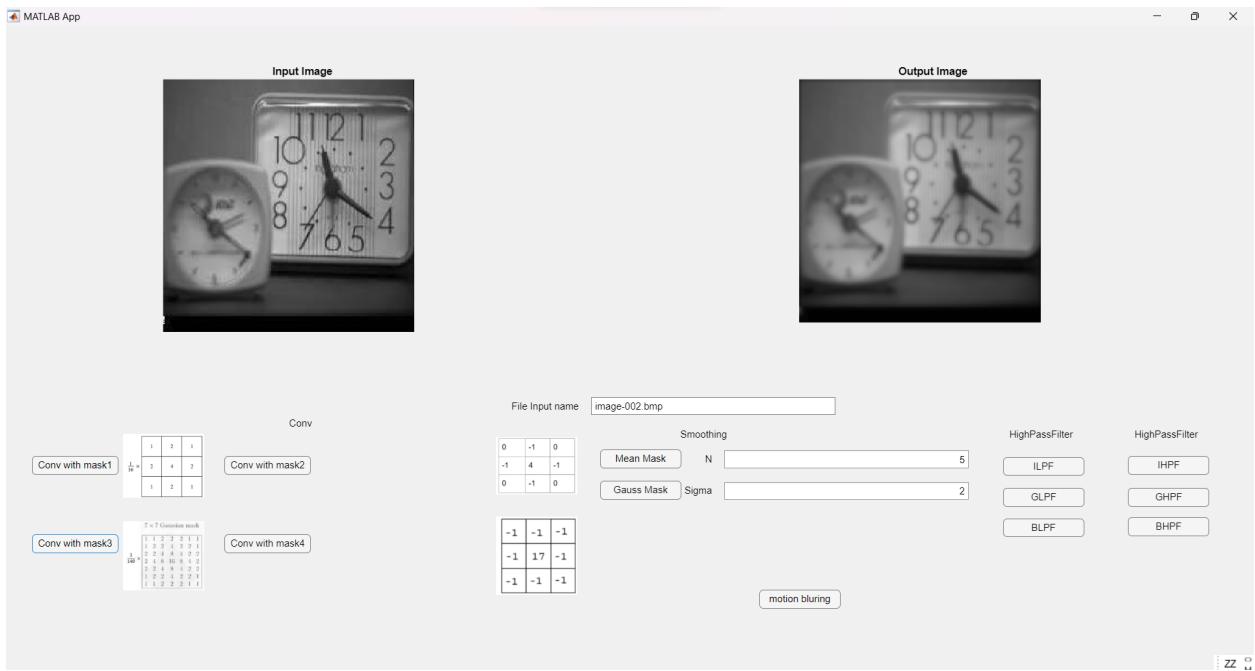
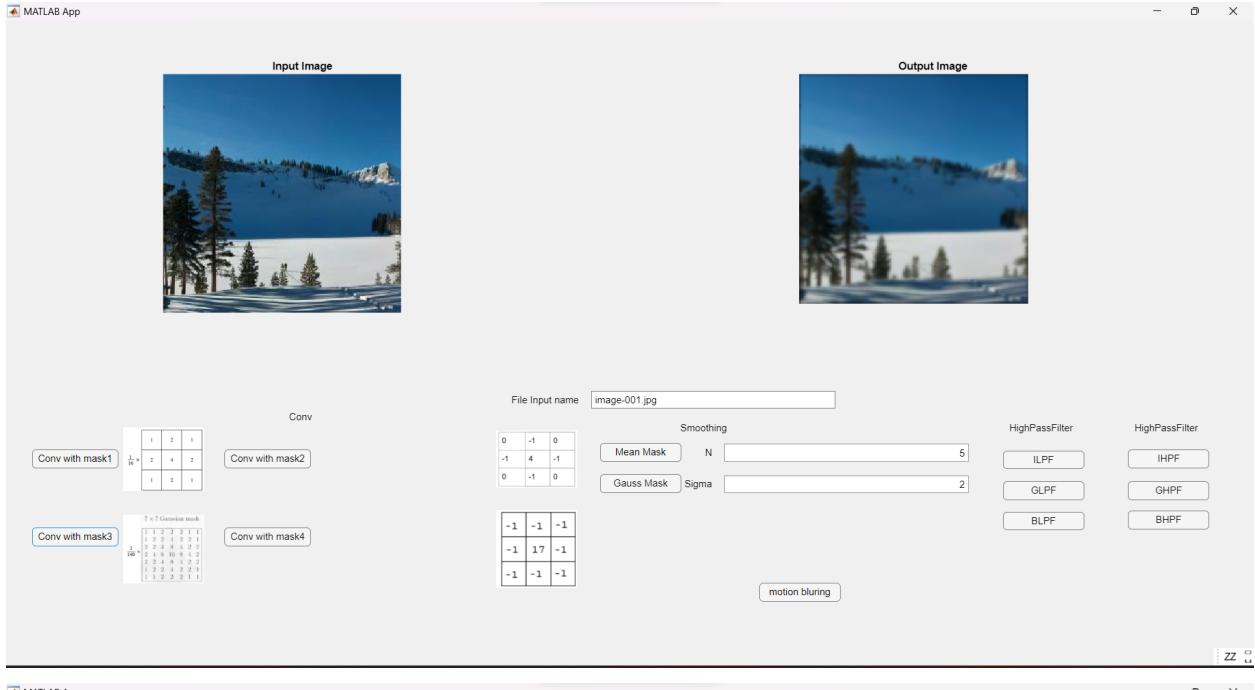


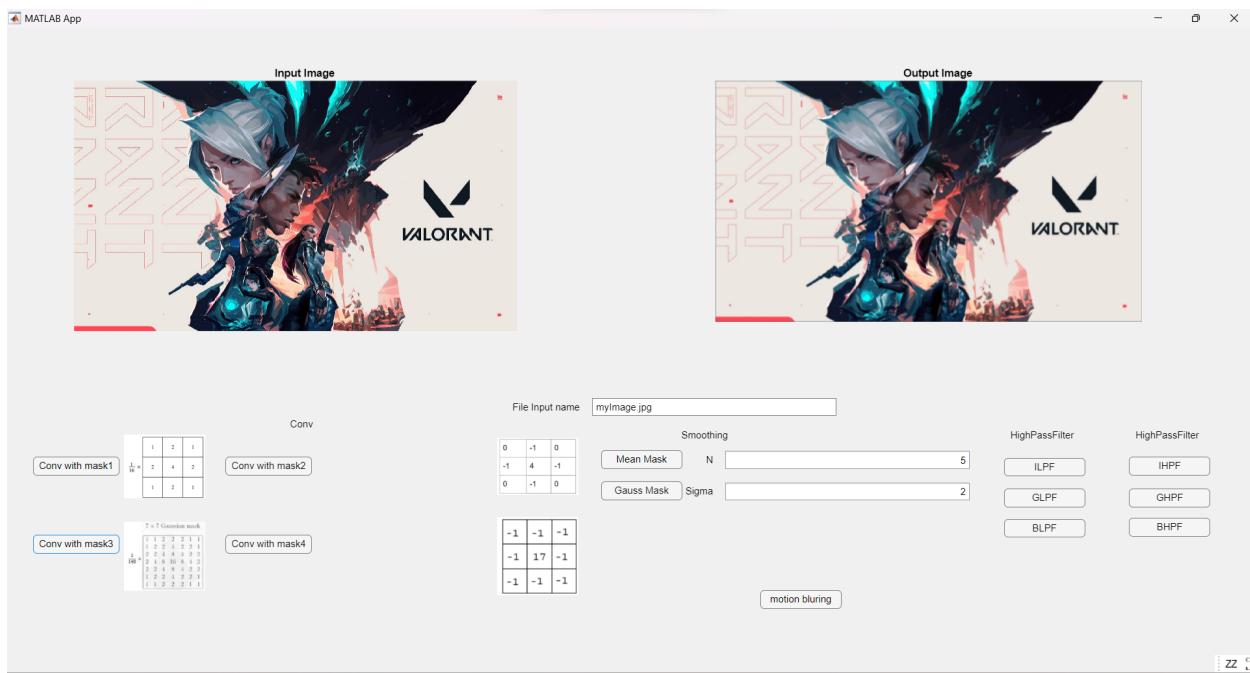
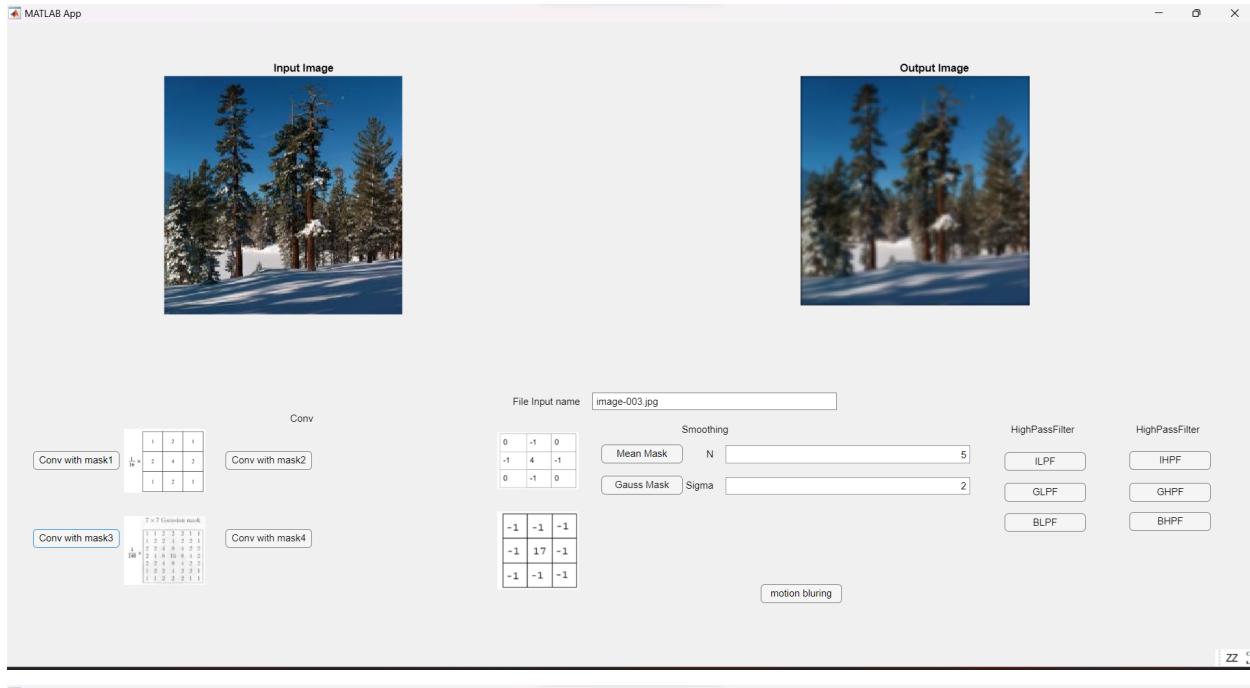
3. Mask 3

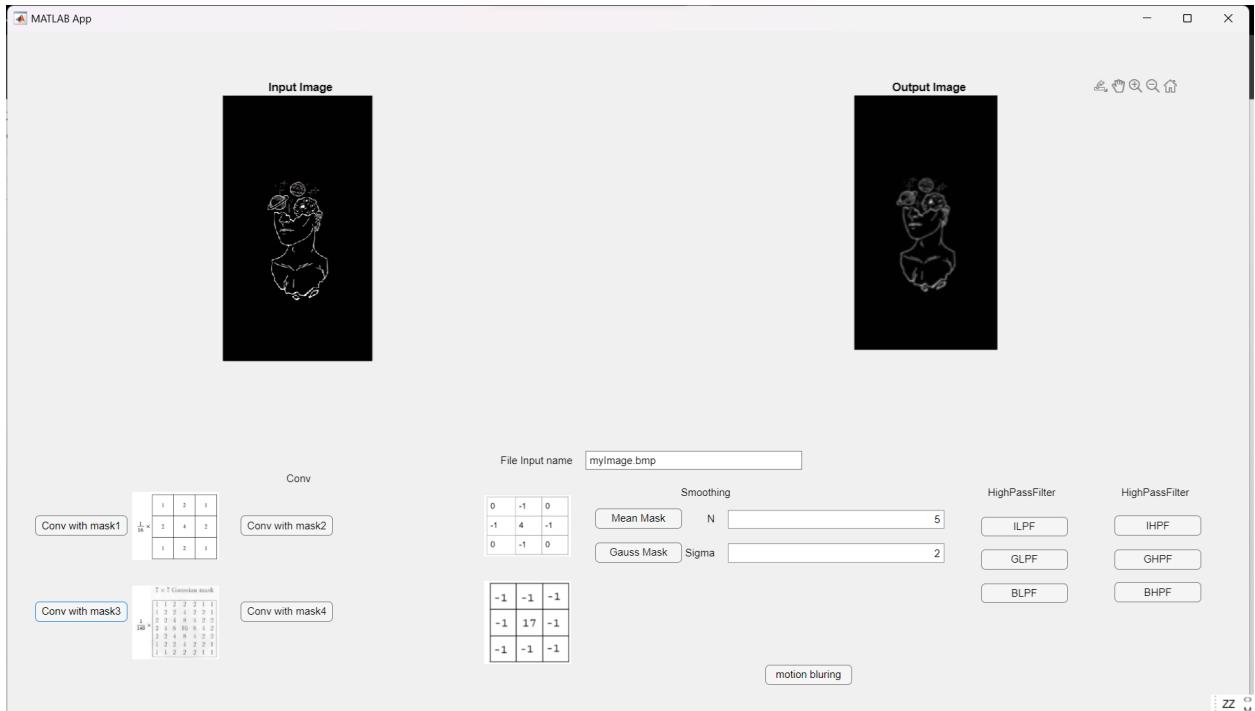
7×7 Gaussian mask

$$\frac{1}{140} \times \begin{bmatrix} 1 & 1 & 2 & 2 & 2 & 1 & 1 \\ 1 & 2 & 2 & 4 & 2 & 2 & 1 \\ 2 & 2 & 4 & 8 & 4 & 2 & 2 \\ 2 & 4 & 8 & 16 & 8 & 4 & 2 \\ 2 & 2 & 4 & 8 & 4 & 2 & 2 \\ 1 & 2 & 2 & 4 & 2 & 2 & 1 \\ 1 & 1 & 2 & 2 & 2 & 1 & 1 \end{bmatrix}$$



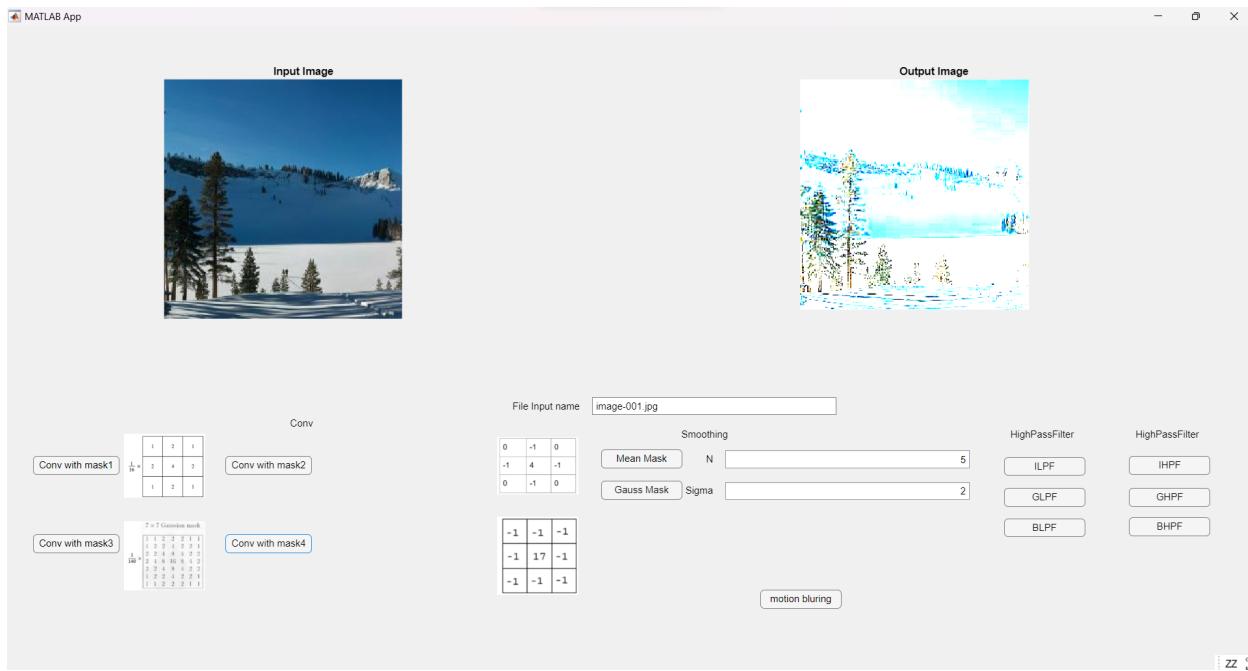
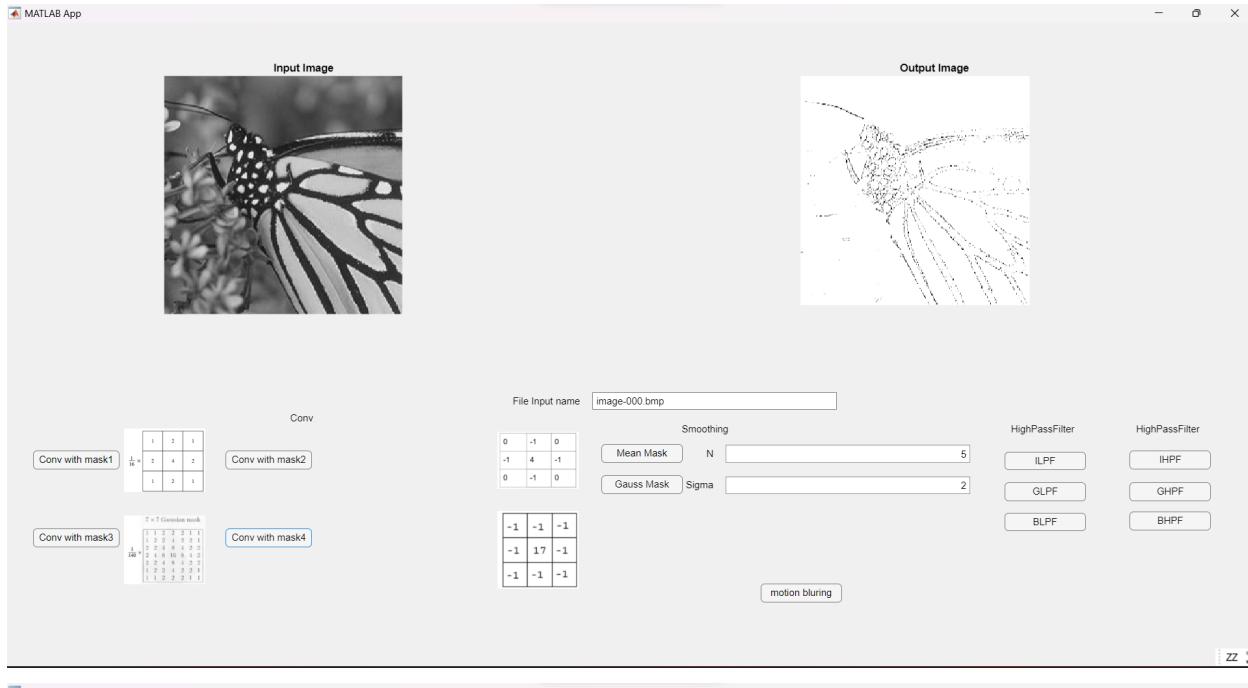


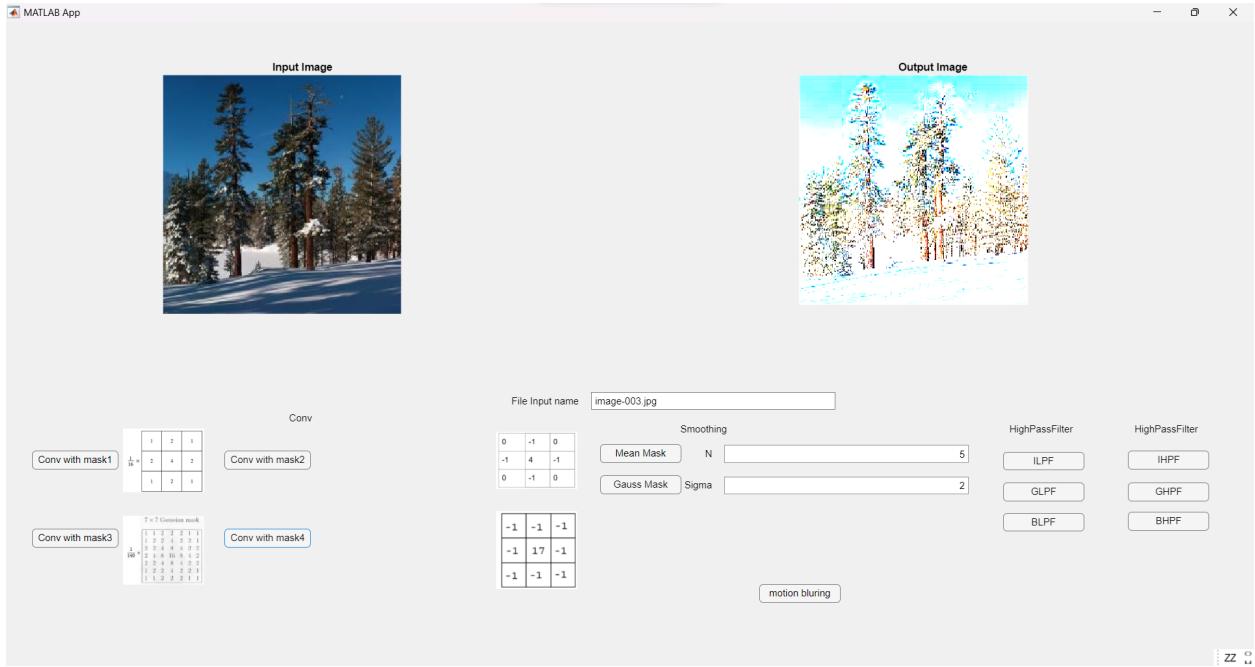
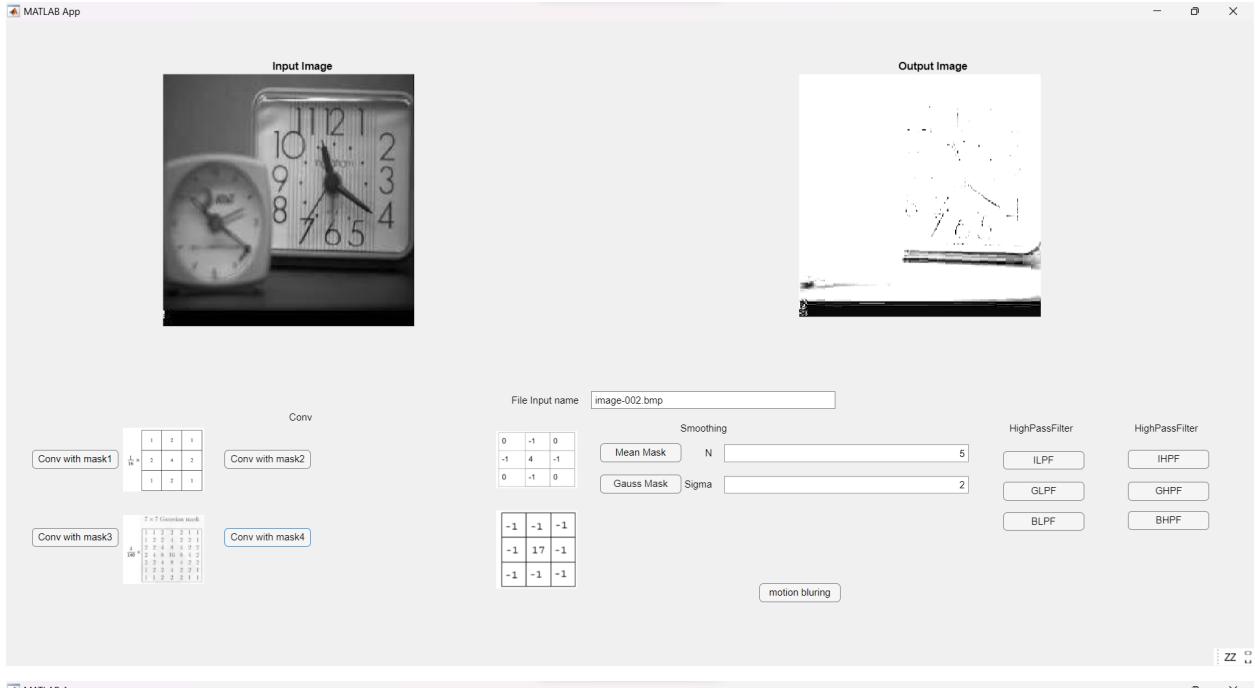


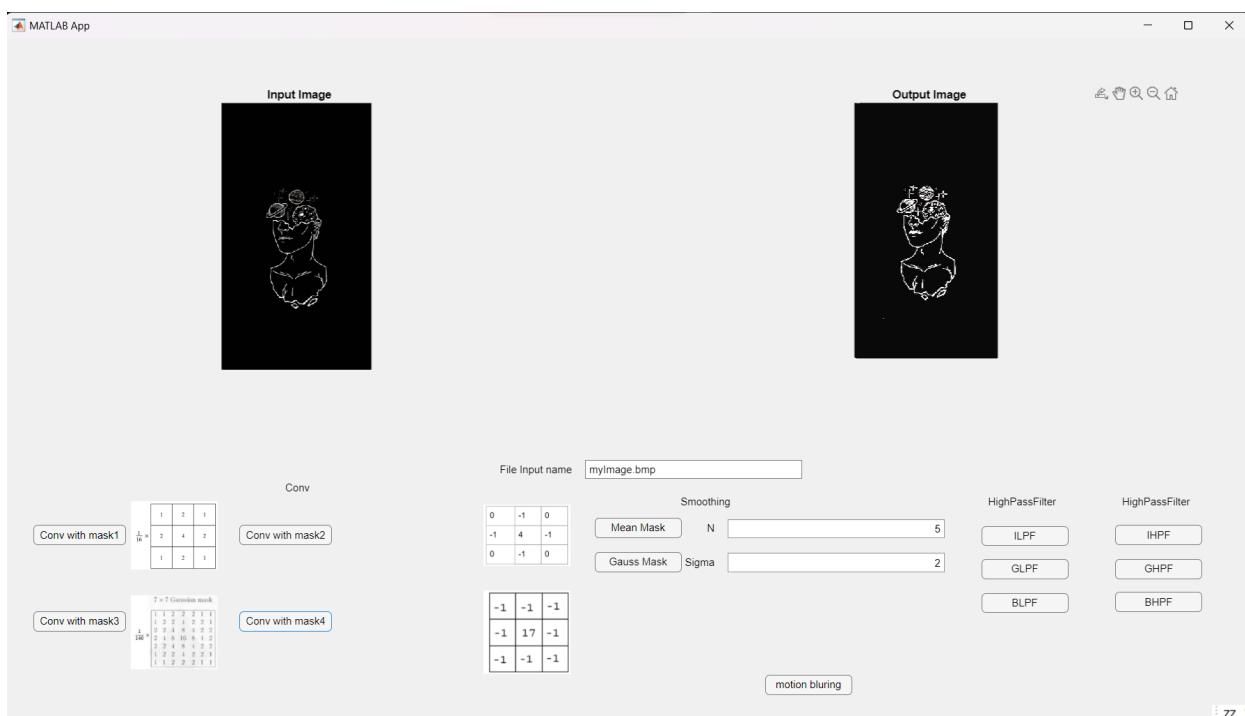
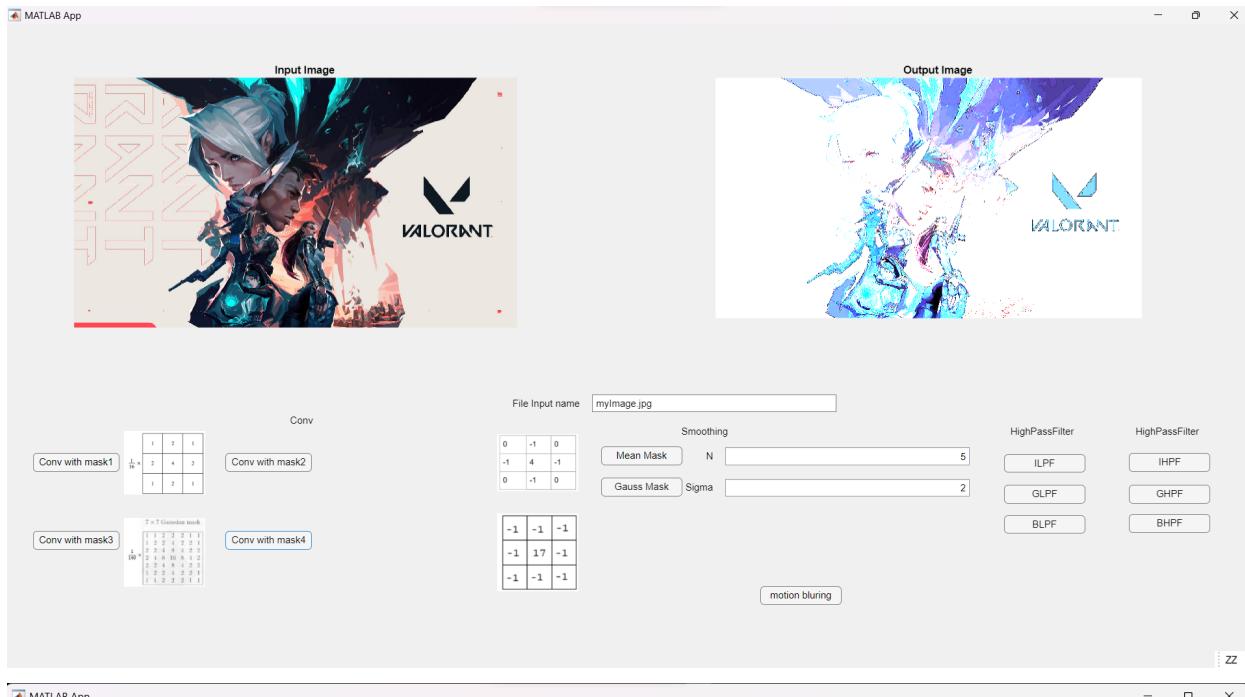


4. Mask 4

-1	-1	-1
-1	17	-1
-1	-1	-1







iii. Analisa

- Mask 1 adalah mask yang menghasilkan efek blurring
- Mask 2 adalah mask yang menghasilkan efek edge detection 4 sisi
- Mask 3 adalah mask gaussian dan menghasilkan efek blur
- Mask 4 adalah masil yang menghasilkan efek edge detection namun diperlukan edge dari semua arah

b. Smoothing dan LowPass Filter

i. Kode Program

```
% BUTTON pushed FUNCTION: MeanMaskButton
function MeanMaskButtonPushed(app, event)
    filename = app.FileInputnameEditField.Value
    img_input = imread(filename);
    imshow(img_input, 'Parent', app.UIAxes);
    n = app.NEditField.Value
    sigma = app.SigmaEditField.Value
    img_result = uint8(image_mean_rgb(double(img_input), n));
    %img_result = uint8(image_gauss_rgb(double(img_input), n, sigma));

    imshow(img_result, 'Parent', app.UIAxes_2);

    function result = image_mean_rgb(image,n)
        mean_mask = ones(n) / (n * n);
        result = my_rgb_conv(image, mean_mask);
    end

    function result = image_gauss_rgb(image,n, sigma)
        % sigma = 2.0; % Change the sigma value as needed
        gaussian_mask = fspecial('gaussian', [n n], sigma);
        result = my_rgb_conv(image, gaussian_mask);
    end

    function result = my_rgb_conv(image, mask)
        [m, n, ~] = size(image); % Get the dimensions and ignore the third dimension (color channels)
        [p, q] = size(mask);
        pad = floor((p - 1) / 2); % Padding
        result = zeros(m, n, 3); % Initialize result for three color channels
        imagePadded = padarray(image, [pad, pad], 0, 'both');

        for c = 1:3
            for i = 1:m
                for j = 1:n
                    region = imagePadded(i:i+p-1, j:j+q-1, c);

                    % Melakukan konvolusi
                    result(i, j, c) = sum(sum(region .* mask));
                end
            end
        end
    end
end
```

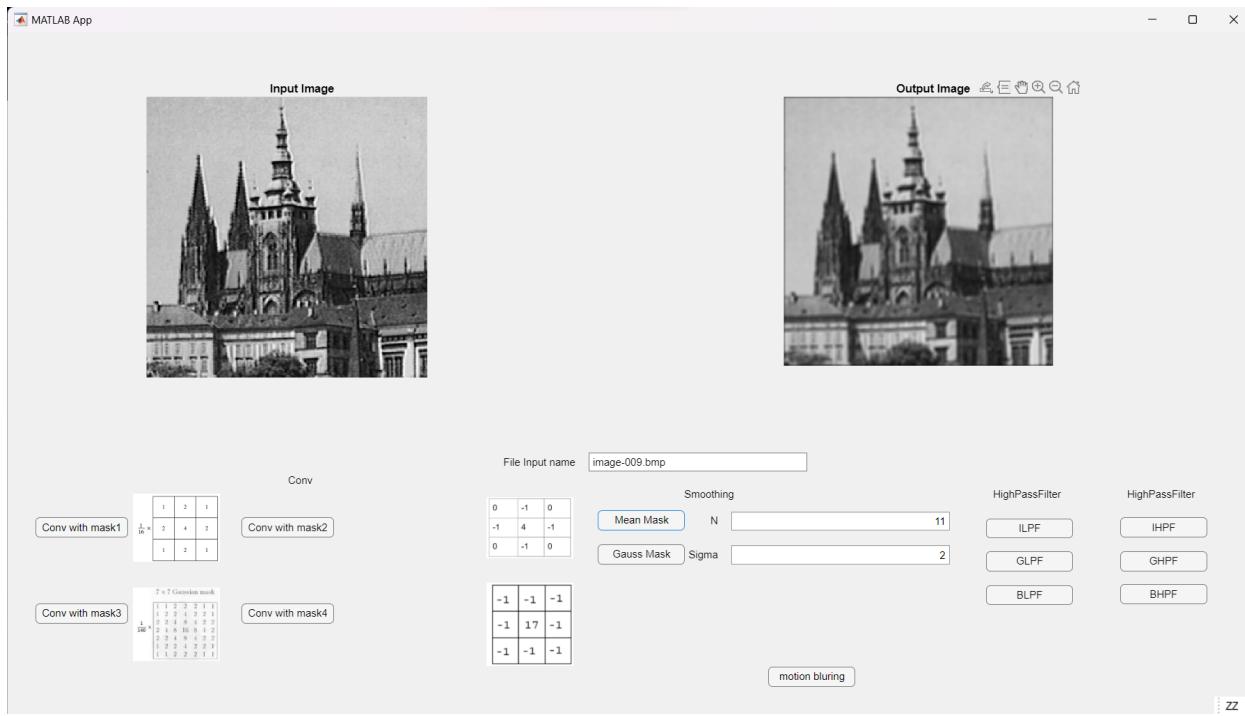
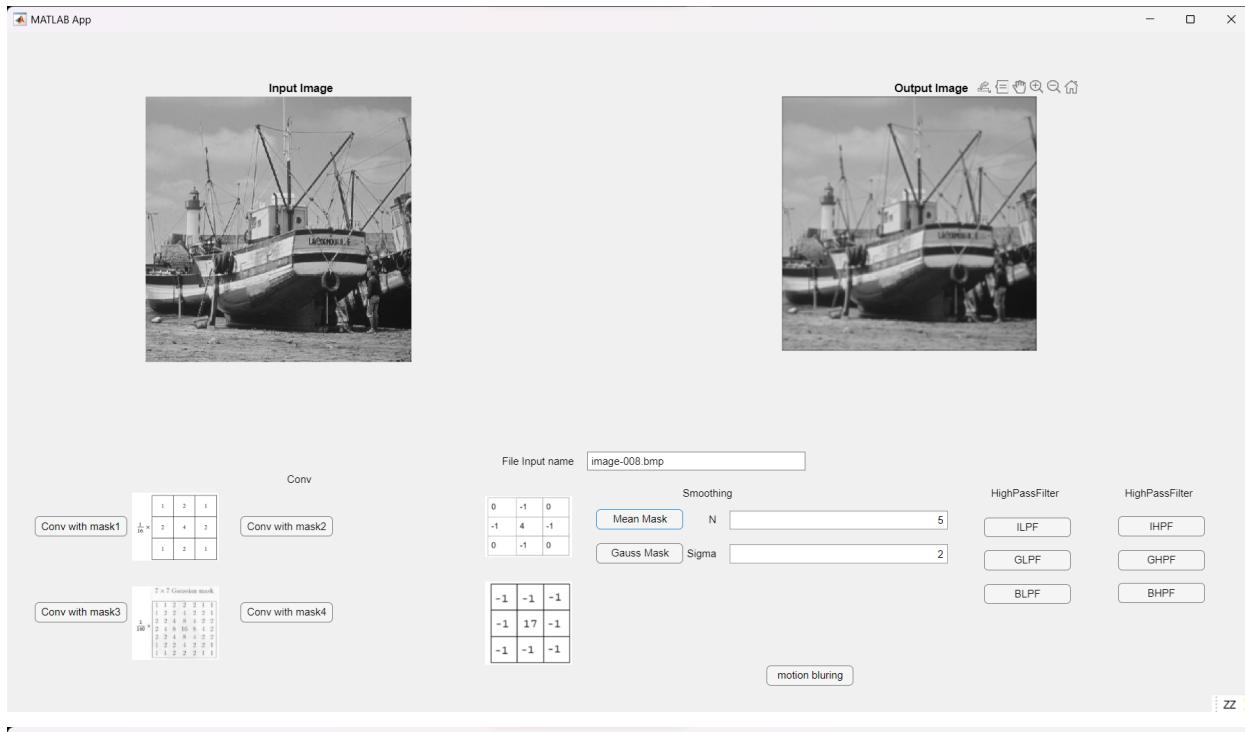
```

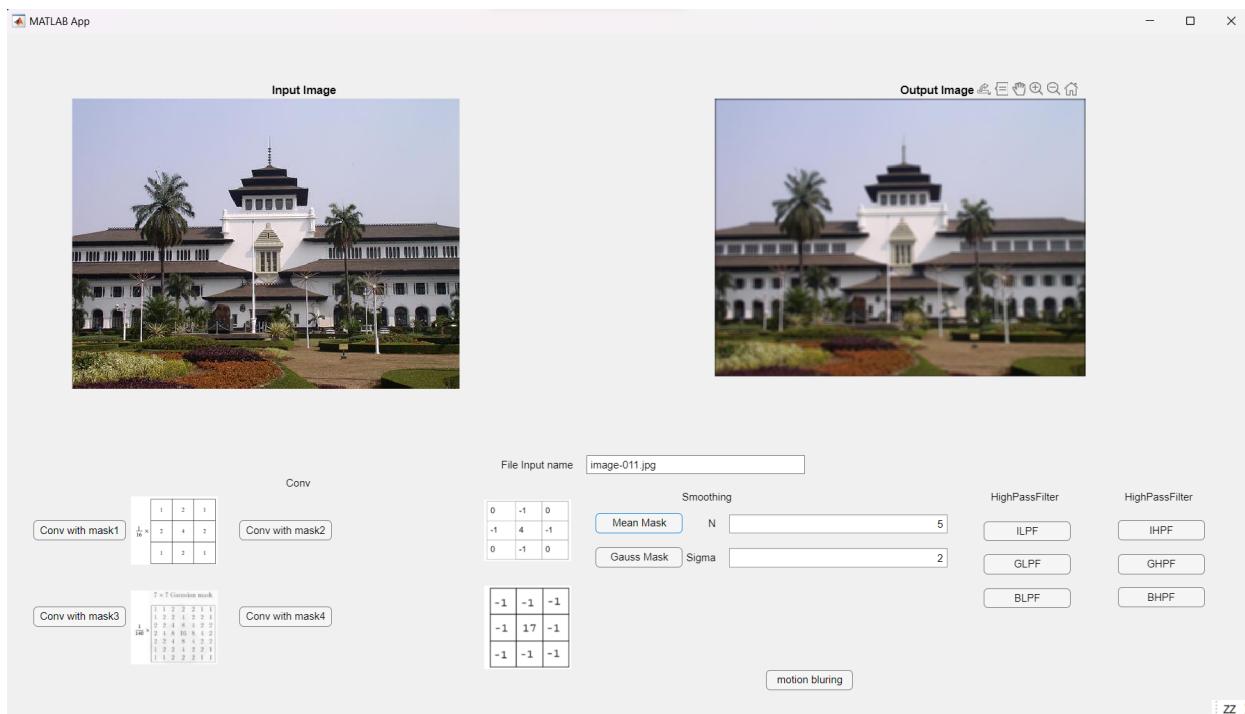
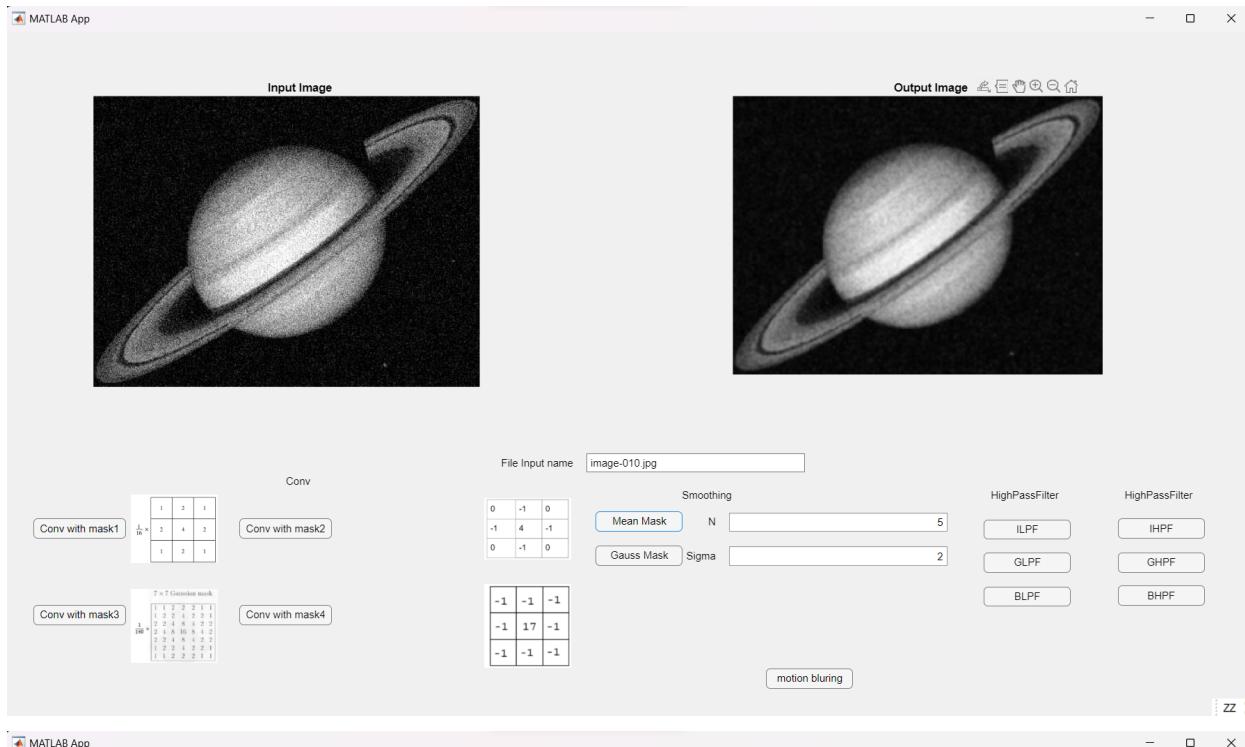
224 % Button pushed function: GaussMaskButton
225 function GaussMaskButtonPushed(app, event)
226     filename = app.FileInputnameEditField.Value
227     img_input = imread(filename);
228     imshow(img_input, 'Parent', app.UIAxes);
229     n = app.NEditField.Value
230     sigma = app.SigmaEditField.Value
231     %img_result = uint8(image_mean_rgb(double(img_input), n));
232     img_result = uint8(image_gauss_rgb(double(img_input), n, sigma));
233
234     imshow(img_result, 'Parent', app.UIAxes_2);
235
236 function result = image_mean_rgb(image,n)
237     mean_mask = ones(n) / (n * n);
238     result = my_rgb_conv(image, mean_mask);
239 end
240
241 function result = image_gauss_rgb(image,n, sigma)
242     % sigma = 2.0; % Change the sigma value as needed
243     gaussian_mask = fspecial('gaussian', [n n], sigma);
244     result = my_rgb_conv(image, gaussian_mask);
245 end
246
247
248 function result = my_rgb_conv(image, mask)
249     [m, n, ~] = size(image); % Get the dimensions and ignore the third dimension (color channels)
250     [p, q] = size(mask);
251     pad = floor((p - 1) / 2); % Padding
252     result = zeros(m, n, 3); % Initialize result for three color channels
253     imagePadded = padarray(image, [pad, pad], 0, 'both');
254
255 for c = 1:3
256     for i = 1:m
257         for j = 1:n
258             region = imagePadded(i:i+p-1, j:j+q-1, c);
259
260             % Melakukan konvolusi
261             result(i, j, c) = sum(sum(region .* mask));
262         end
263     end
264 end
265
266 end
267 end
268 end

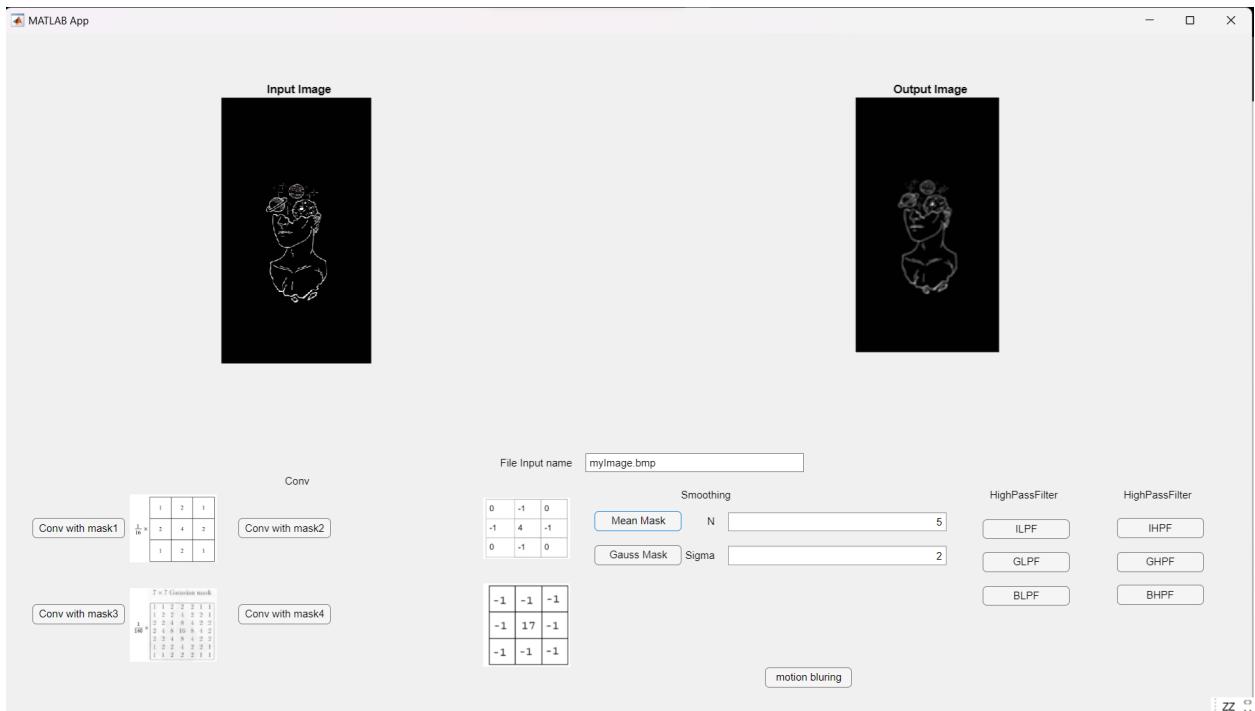
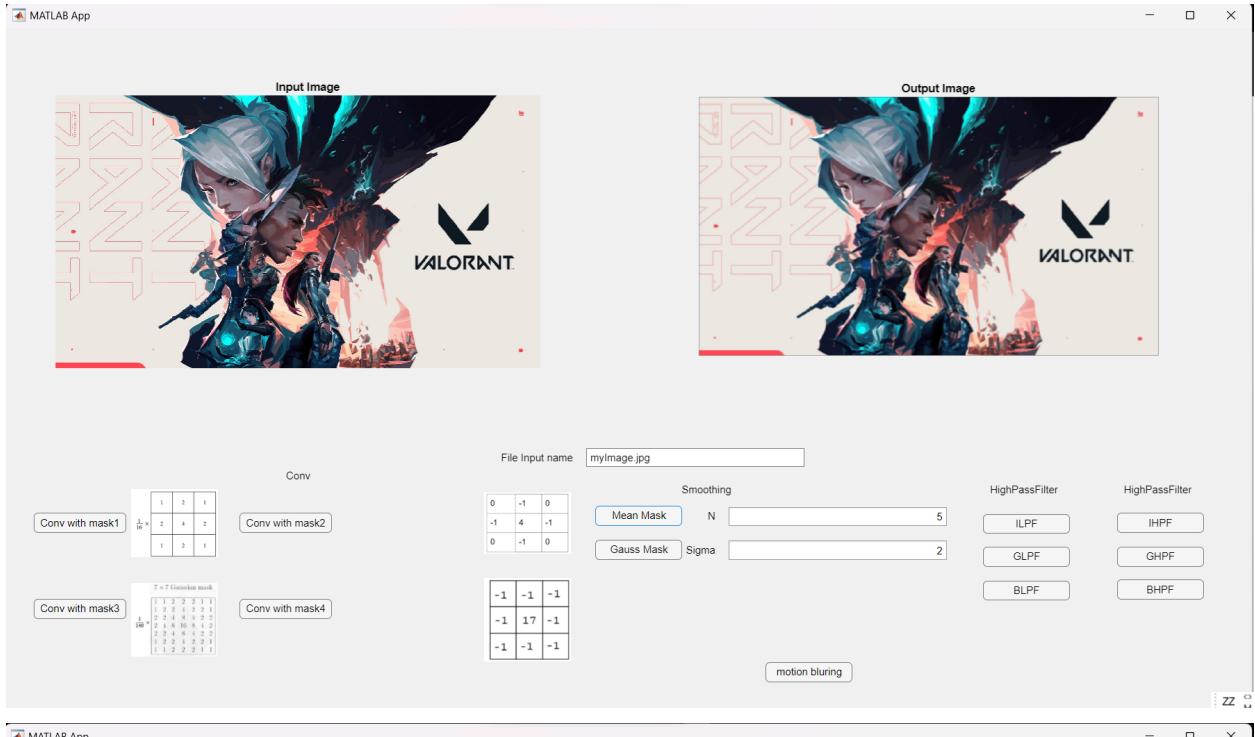
```

Cat : apabila tombol gauss yang di tekan akan menggunakan fungsi image_gauss

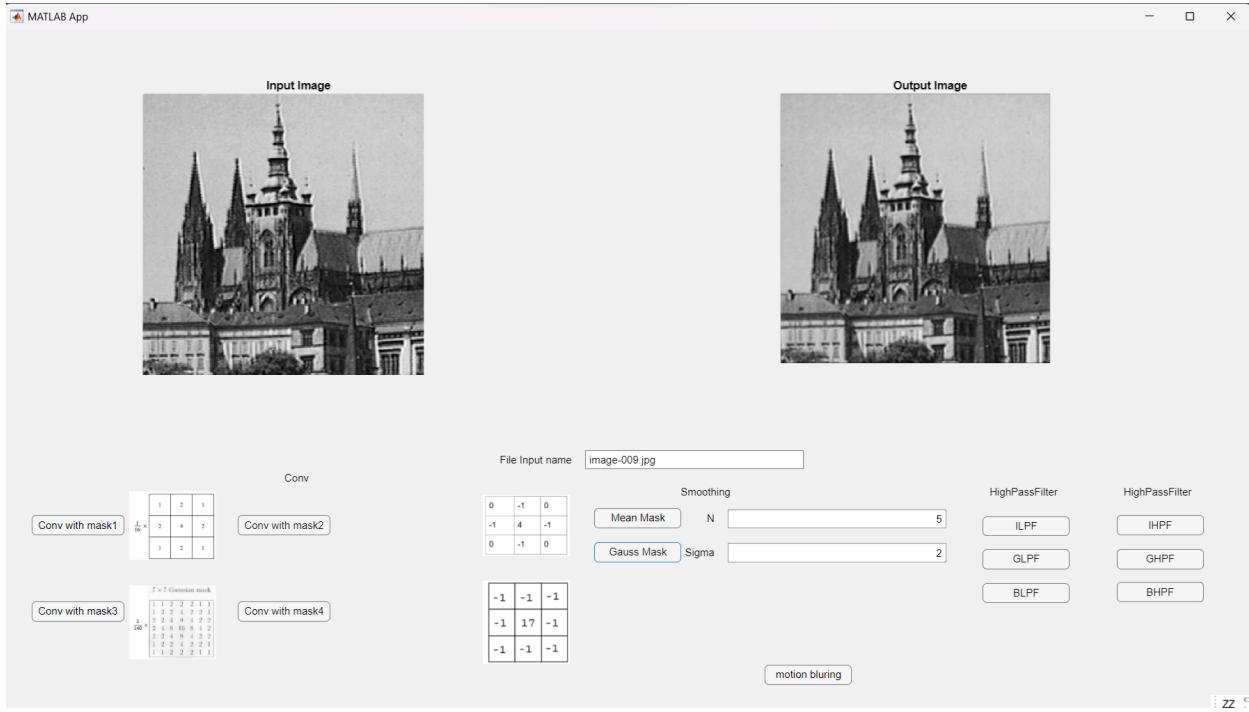
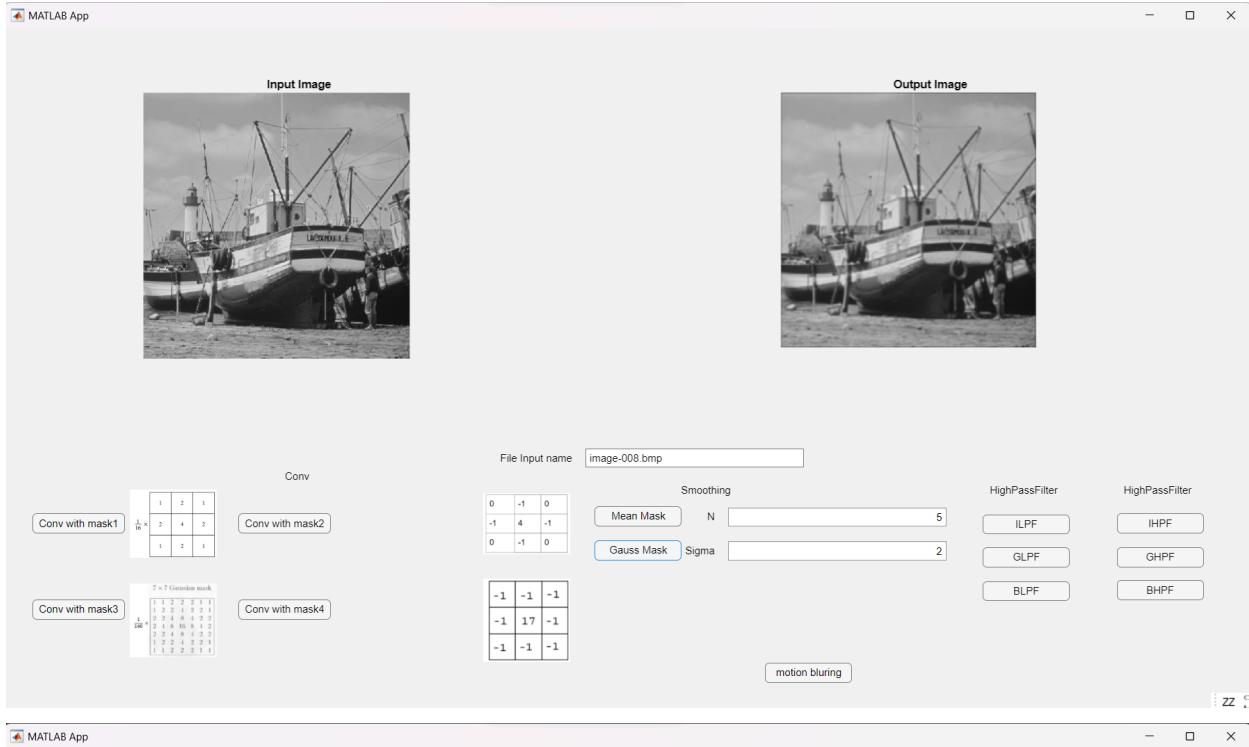
- ii. Hasil Eksekusi
 1. Mean Filter

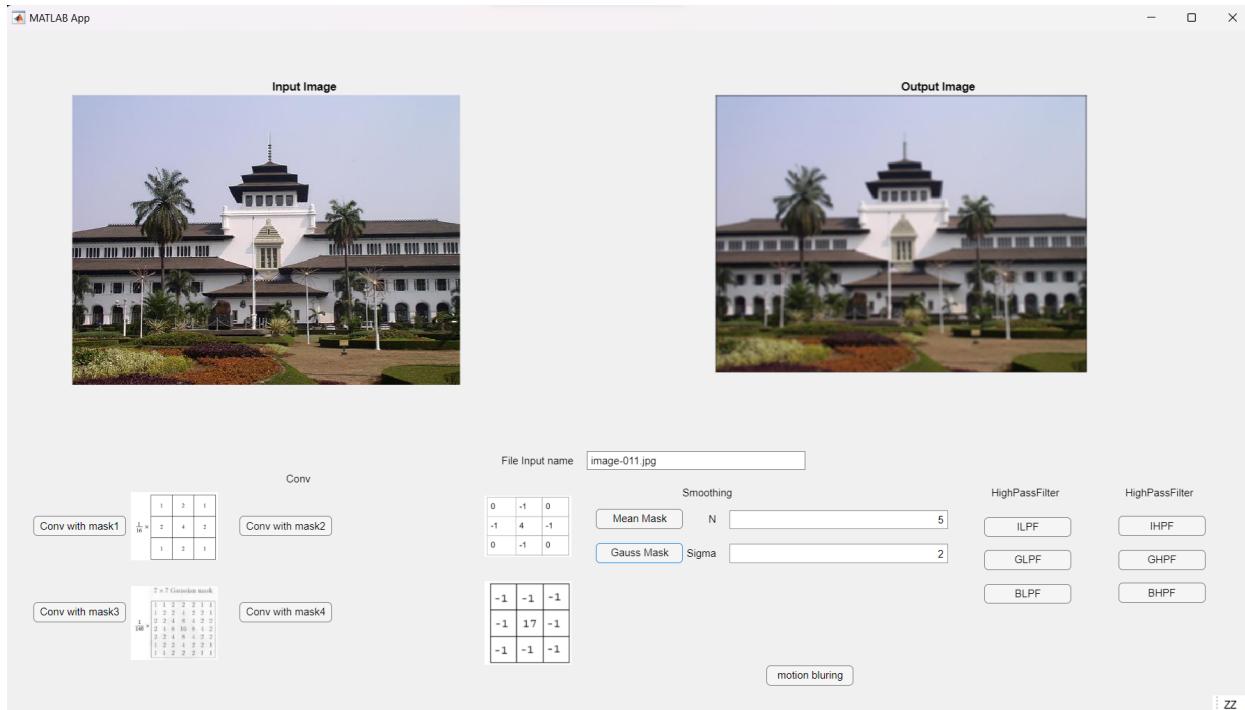
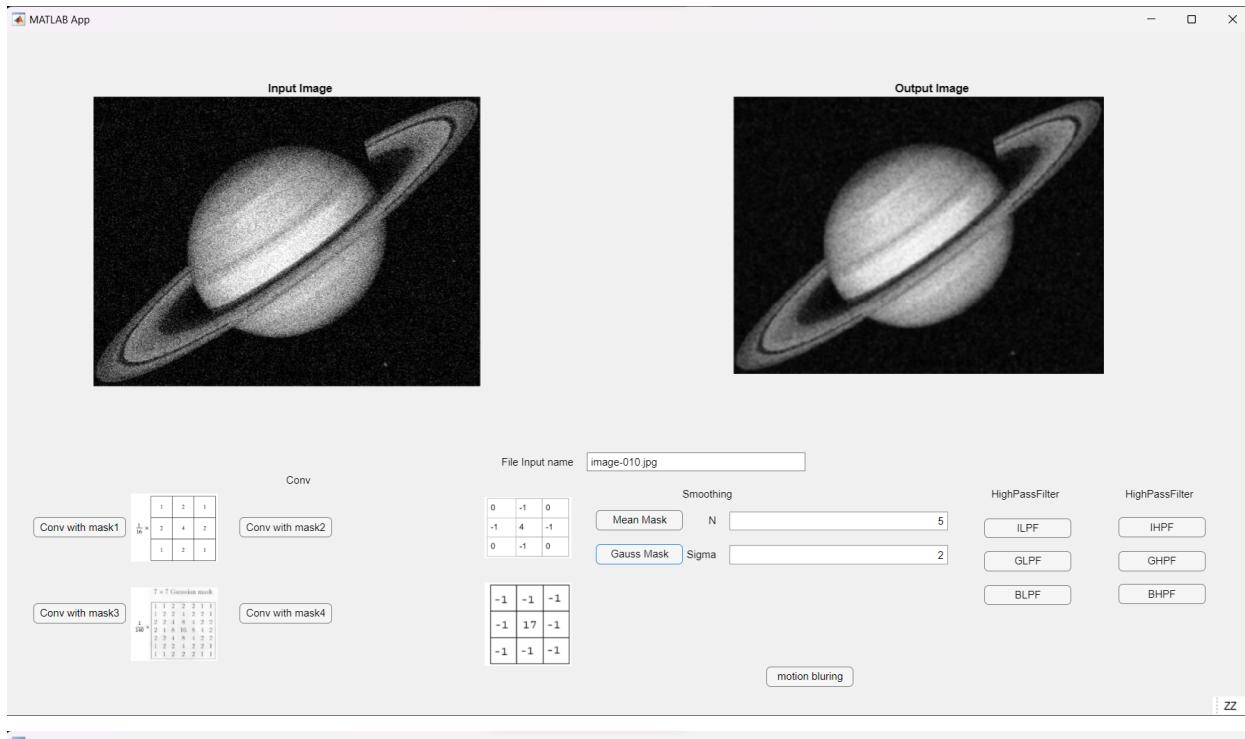


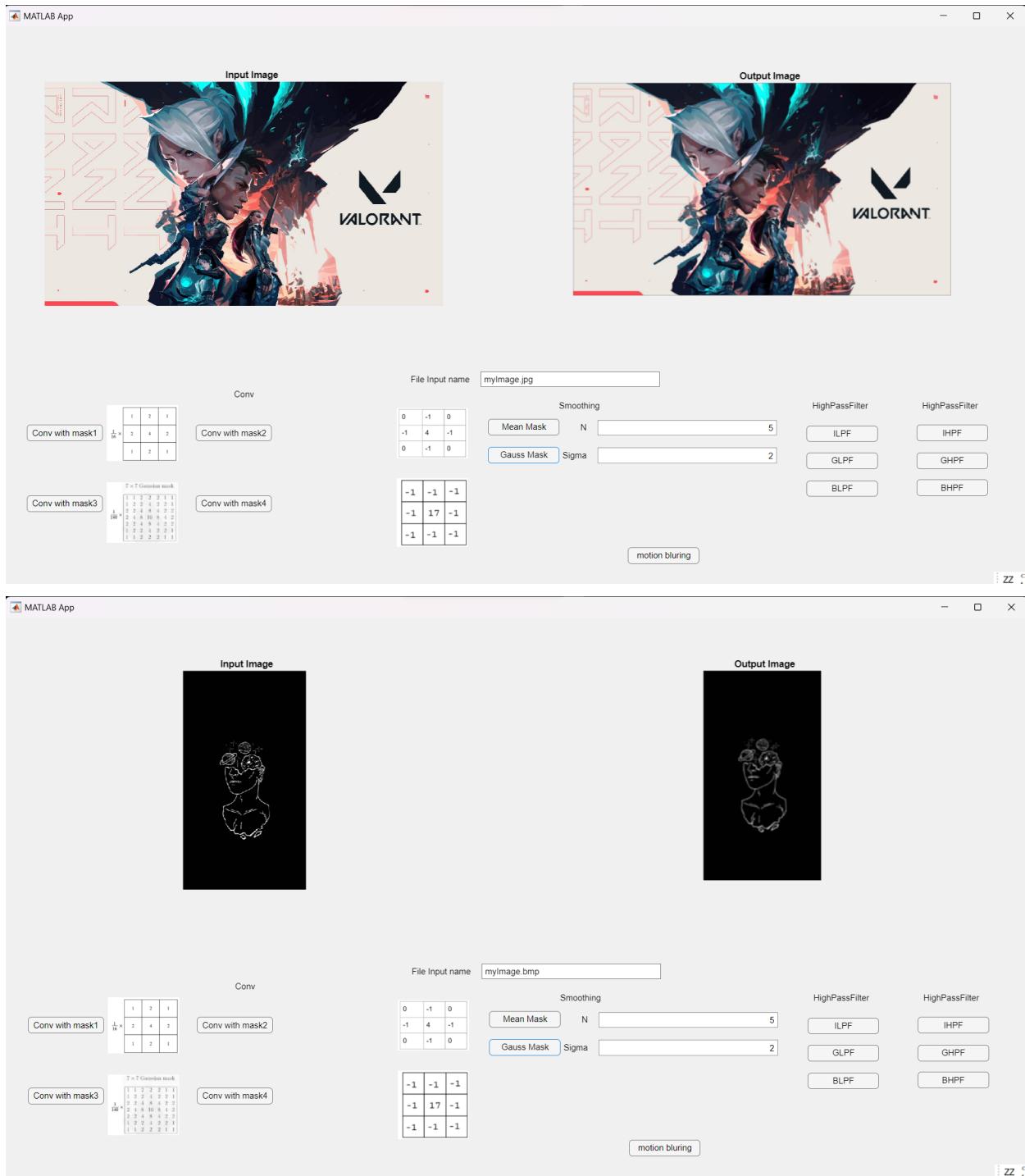




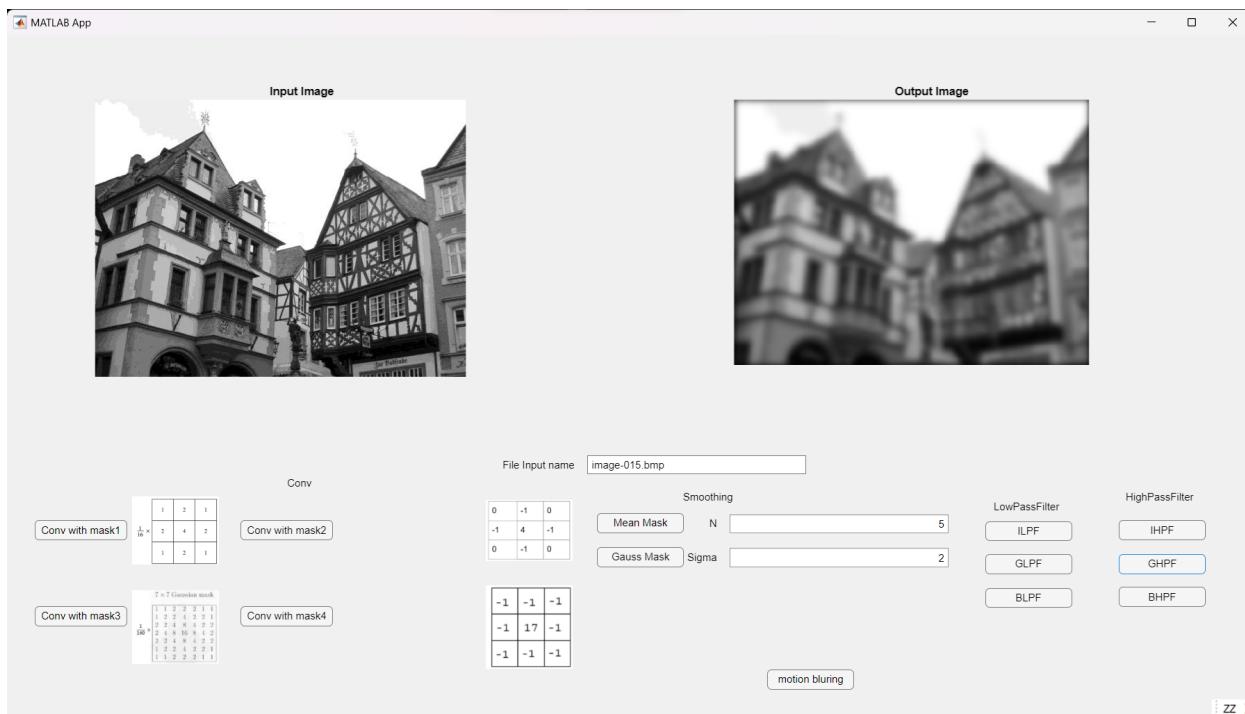
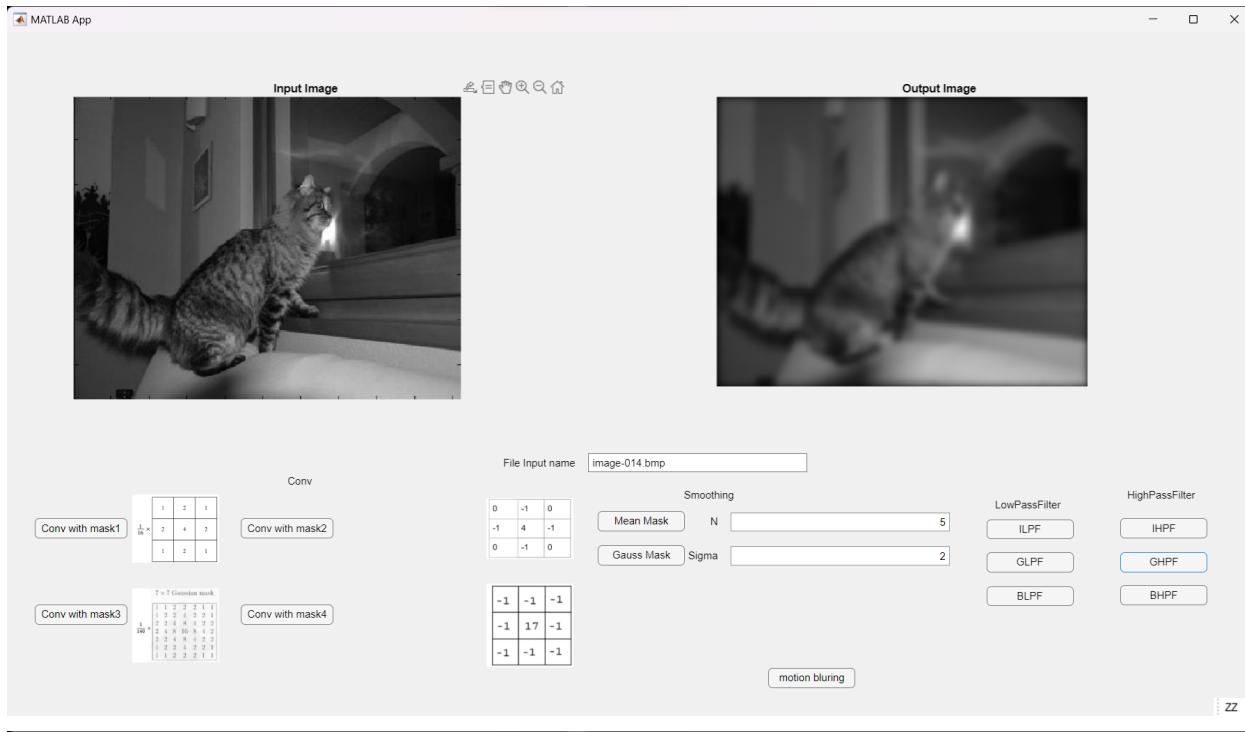
2. Gauss Filter

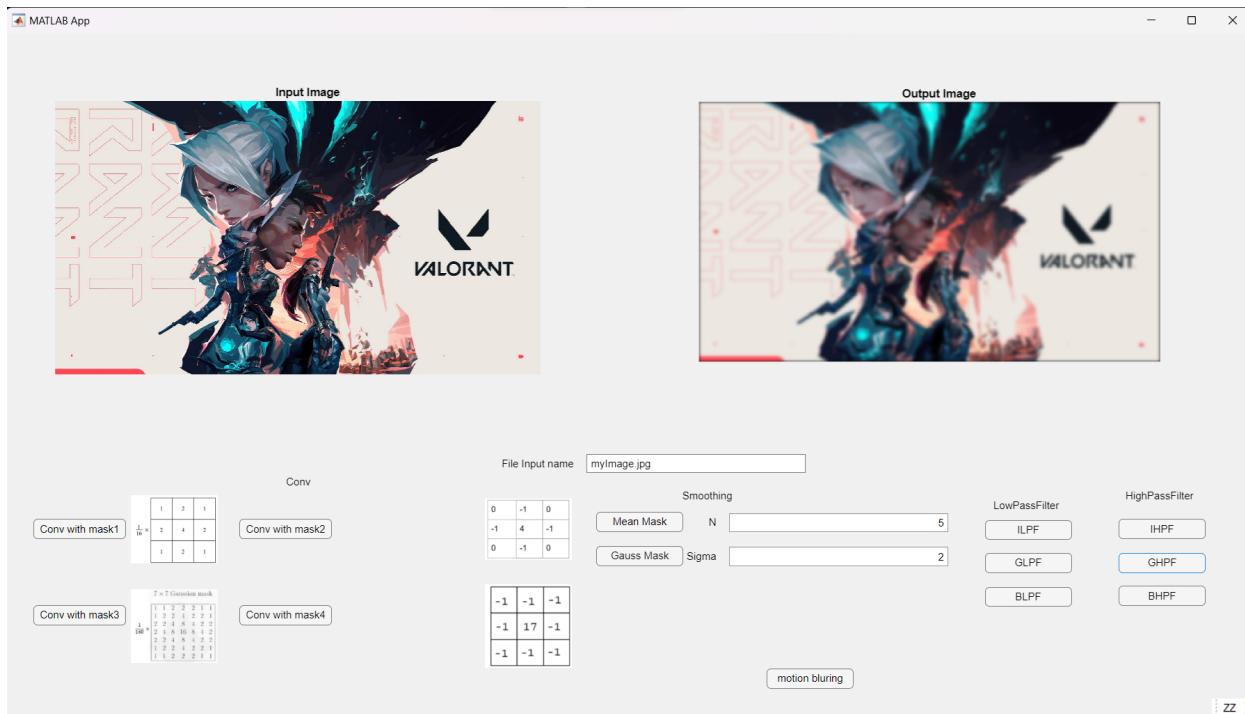
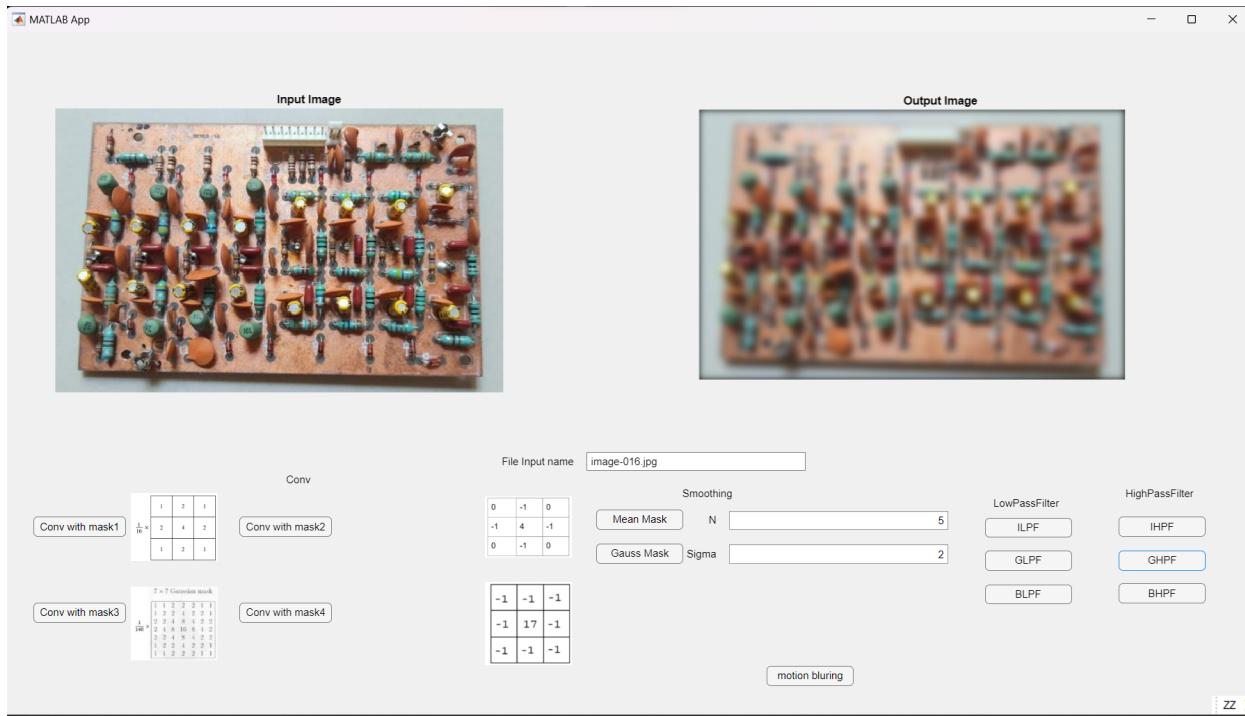


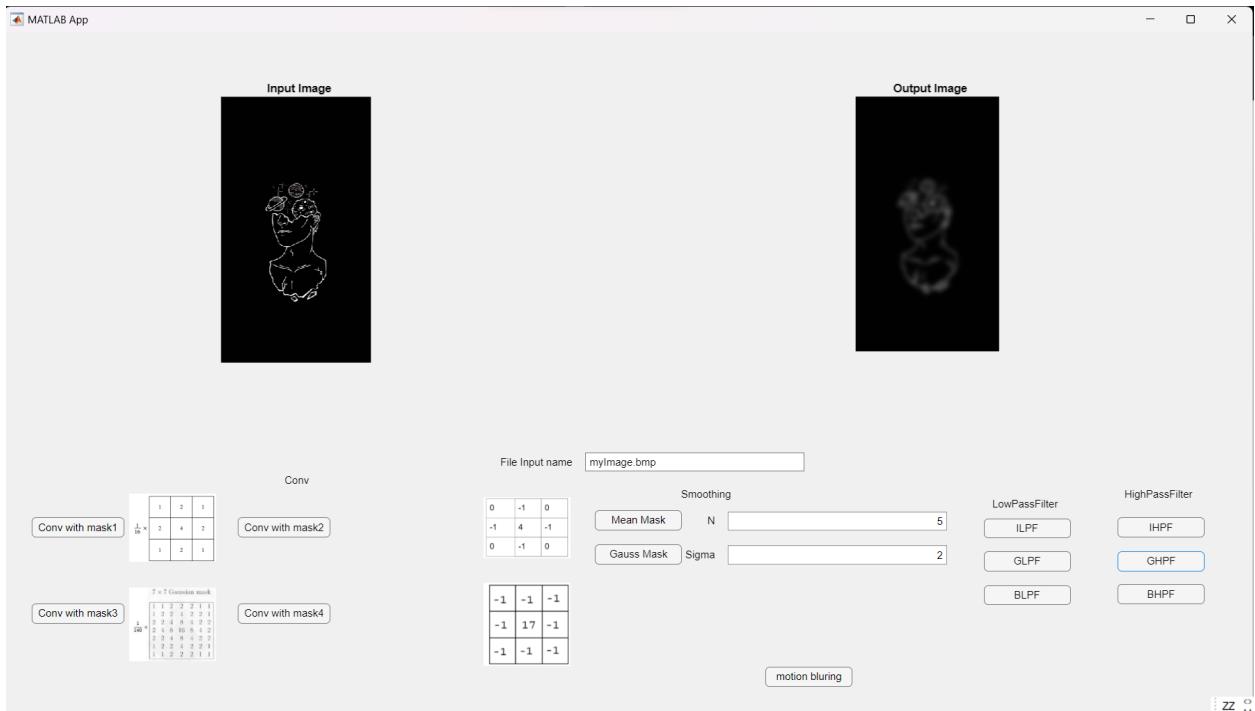




3. Low Pass Filter







iii. analisa

Kedua Mean mask, Gauss mask, dan low pass filter memiliki efek blurring namun pada kernel gauss pixel tengah kenel yang juga pixel asli pada posisi tersebut memiliki efek yang lebih besar daripada pixel di sekitarnya. Karena efek blurring ini juga kernel ini dapat menghilangkan beberapa jenis noise

c. Penapisan citra dalam ranah frekuensi (high-pass filter IHPF, GHPF, dan BHPF)

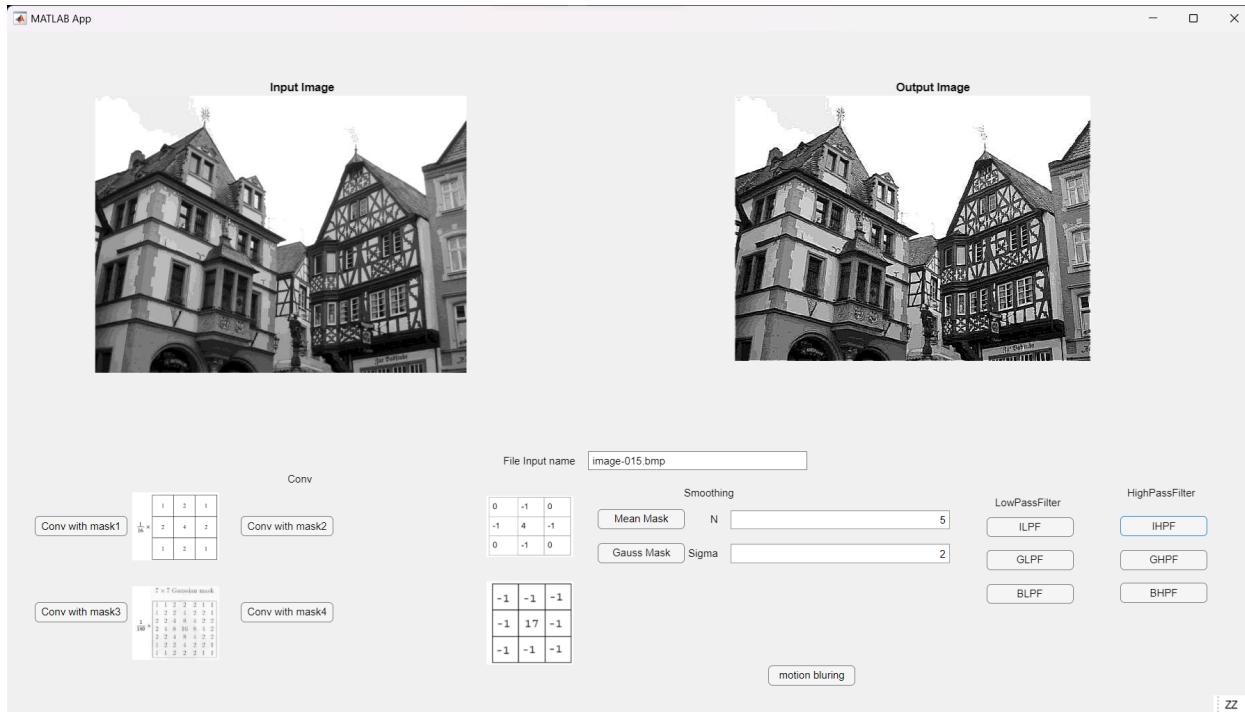
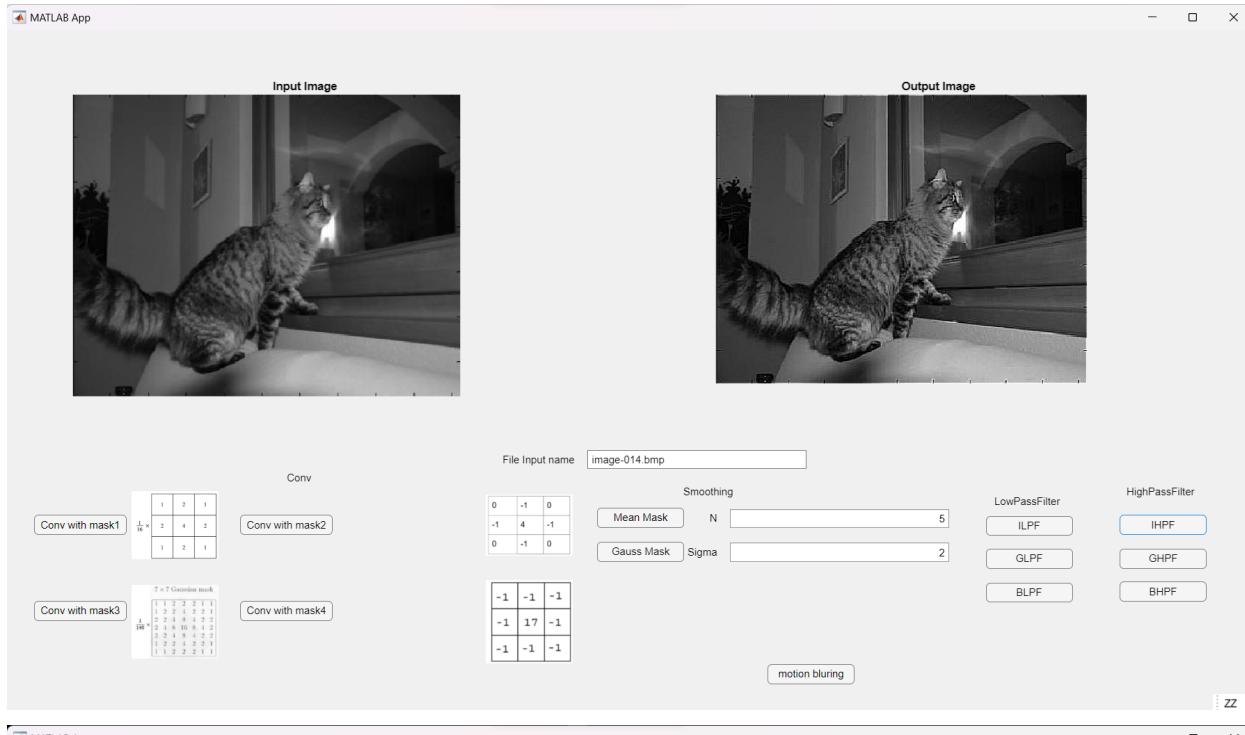
i. Kode Program

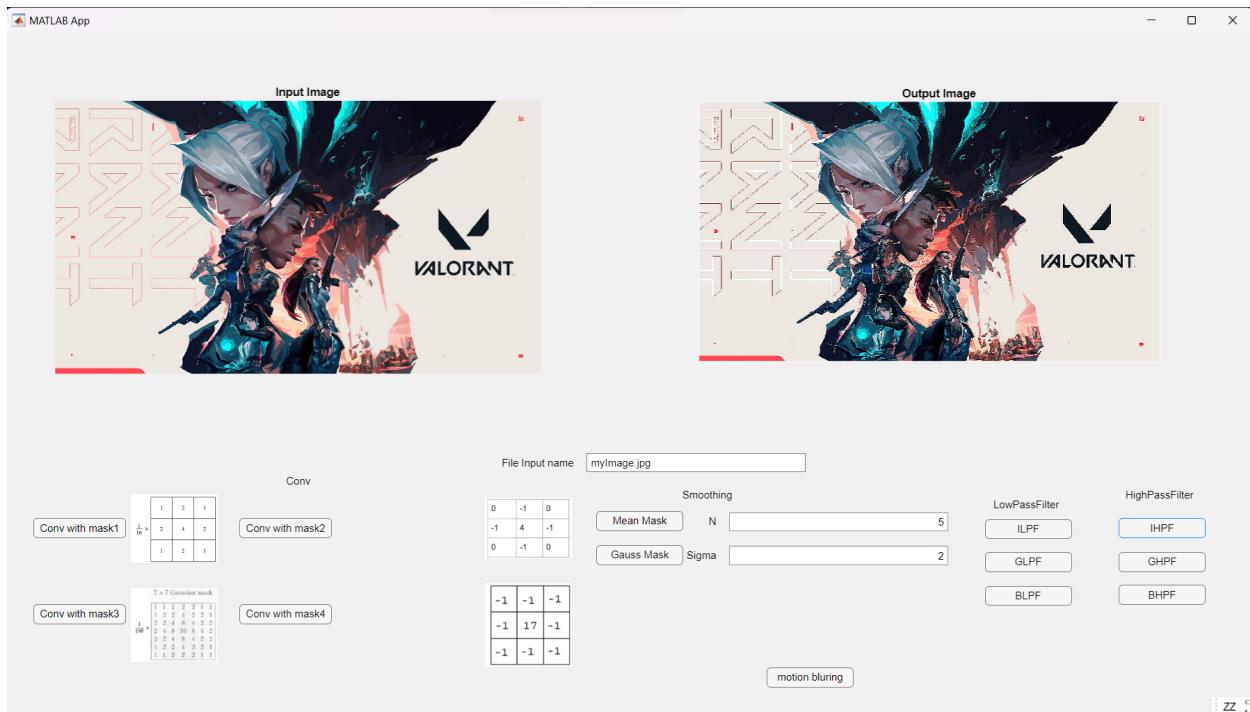
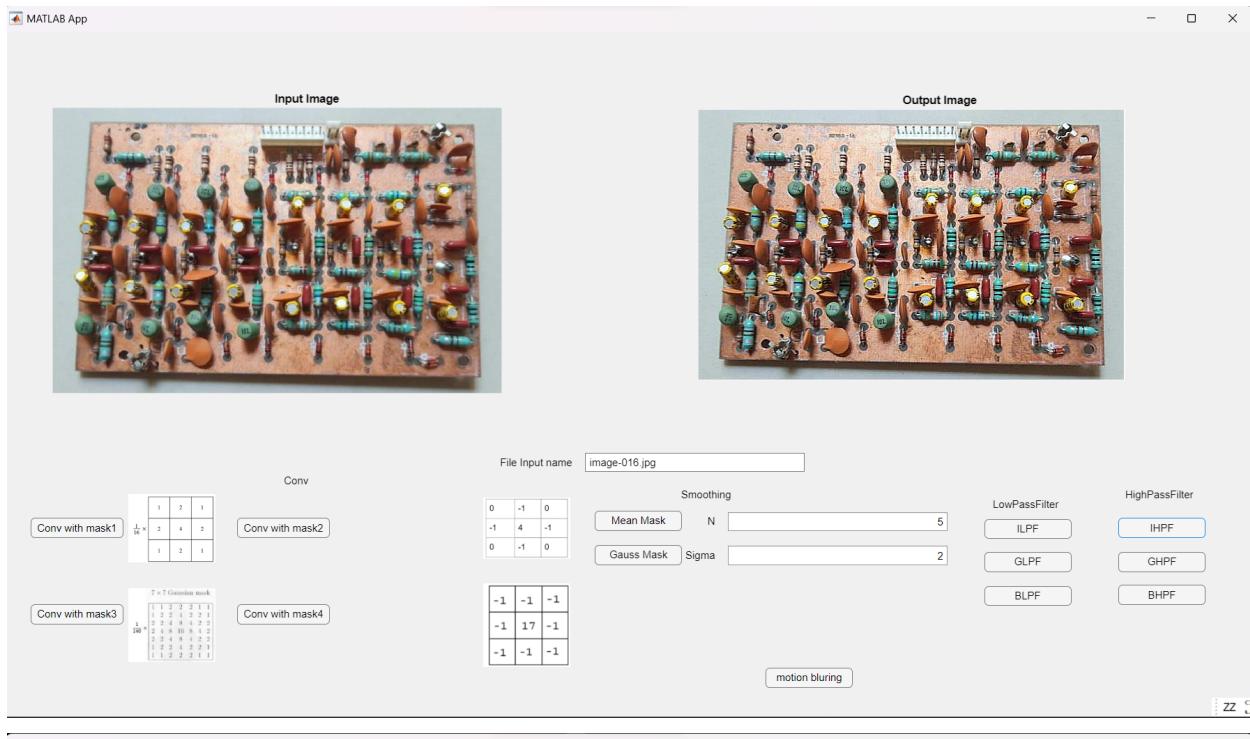
```

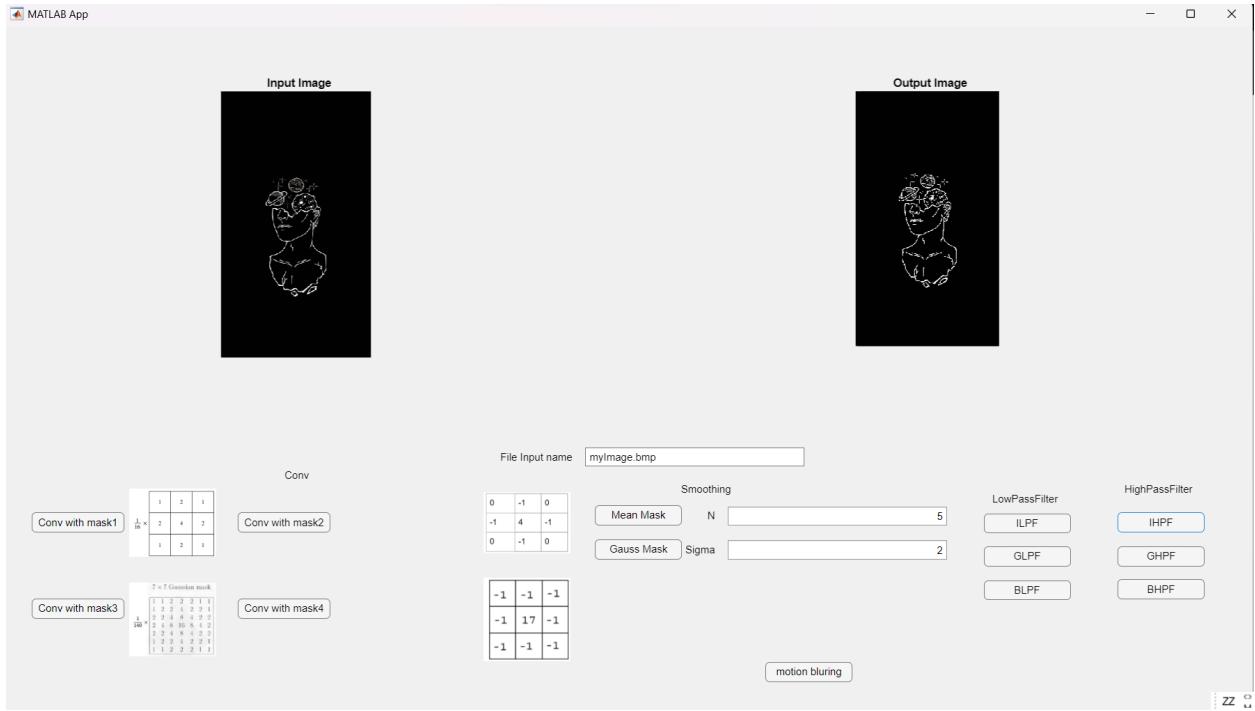
271 % BUTTON pushed function. INPUTBUTTON
272     function IHPFButtonPushed(app, event)
273         % IHPF
274         filename = app.FileInputnameEditField.Value
275         img_input = imread(filename);
276         imshow(img_input, 'Parent', app.UIAxes);
277         ihpf_filter = fspecial('unsharp', 0.5);
278
279         img_result = imfilter(img_input, ihpf_filter);
280
281         imshow(img_result, 'Parent', app.UIAxes_2);
282
283     end

```

ii. Hasil Eksekusi







1. Analisa

High Pass Filter (IHPF, GHPF, BHPF) memiliki efek edge/image sharpening. Dengan BHPF dengan threshold yang lebih gradual memiliki hasil yang paling bagus

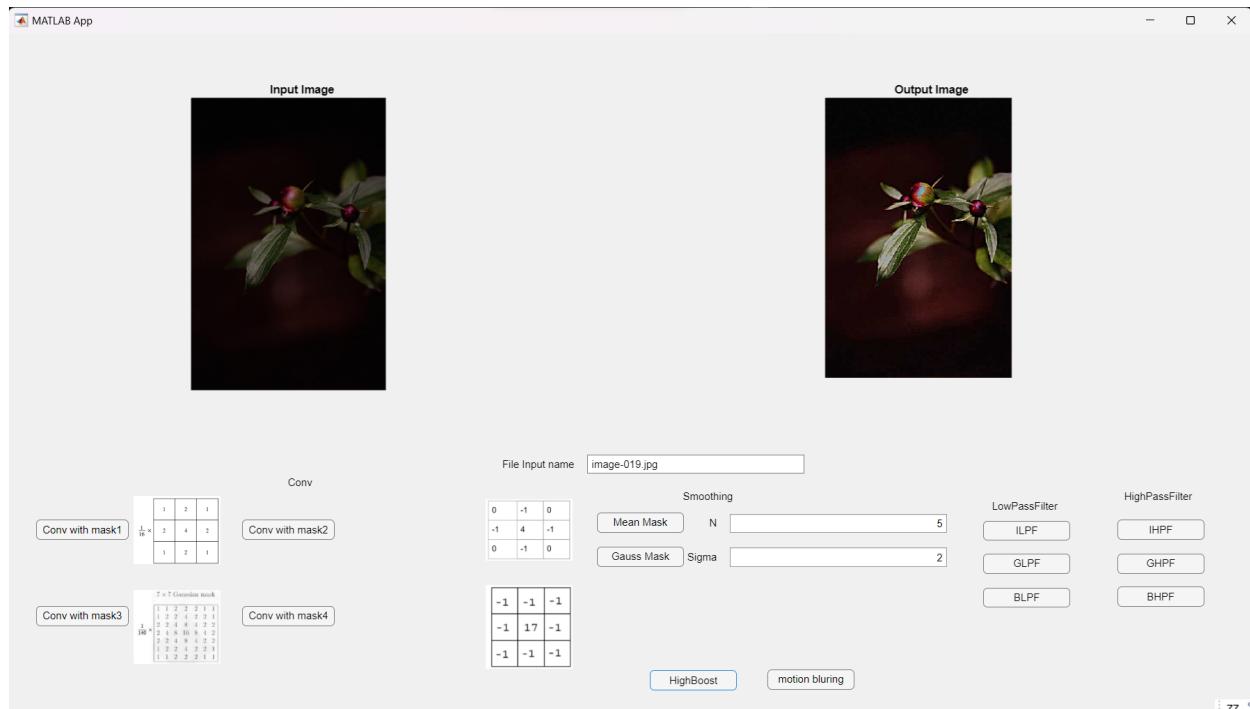
- d. High Frequency Boost Filtering (Penapisan citra dalam ranah frekuensi untuk menghasilkan citra yang lebih terang)
 - i. Kode Program

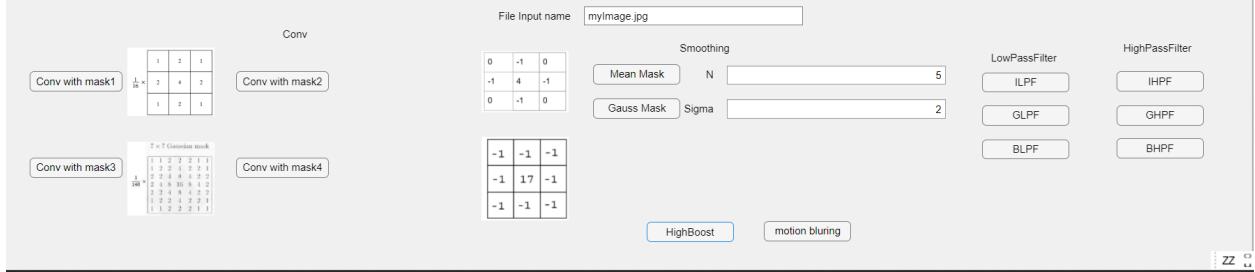
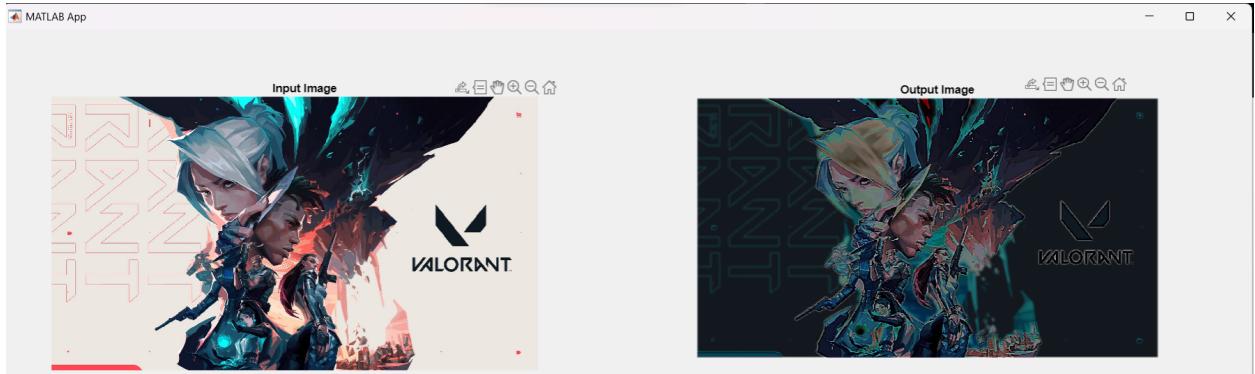
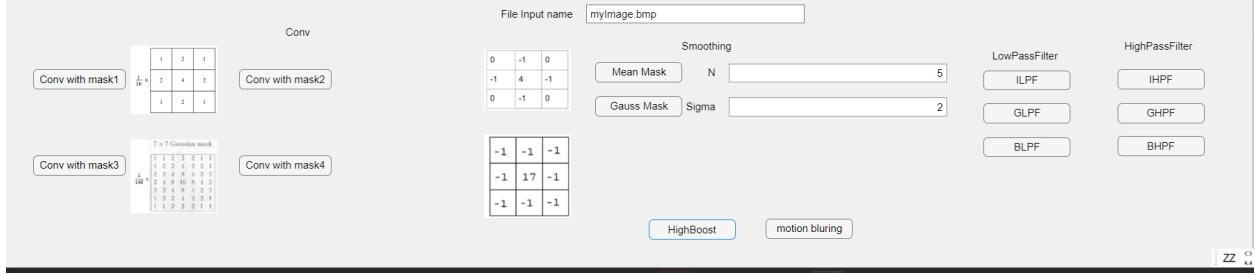
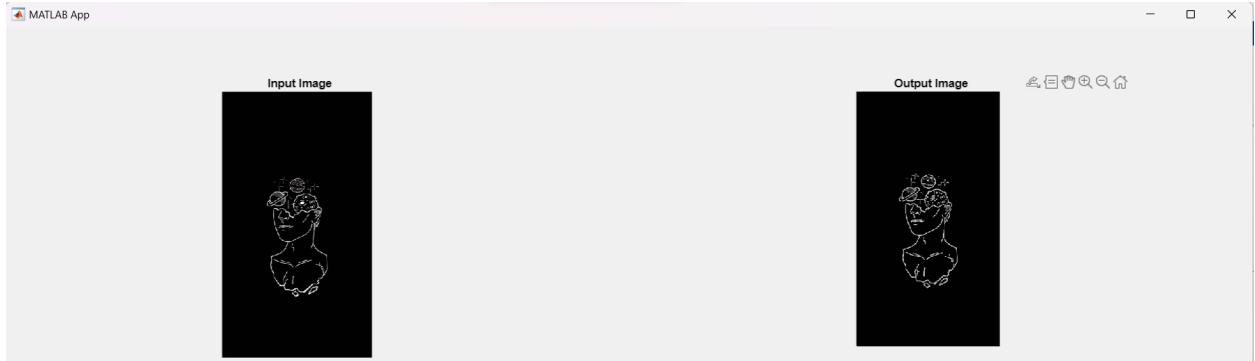
```

333 % Button pushed function: HighBoostButton
334 function HighBoostButtonPushed(app, event)
335
336     % BHPF
337     filename = app.FileInputnameEditField.Value
338     img_input = imread(filename);
339     imshow(img_input, 'Parent', app.UIAxes);
340
341     h_size = 10
342     Sigma = 5
343     A = 2
344     GLPF = fspecial('gaussian', h_size, Sigma);
345     GLPF_Image = imfilter(img_input, GLPF);
346
347     High_pass_filter = img_input - GLPF_Image;
348
349     Unsharpened_Image = (1*img_input) + High_pass_filter;
350
351     HFBF_Image = (A-1)*img_input + img_input - GLPF_Image;
352
353     imshow(HFBF_Image, 'Parent', app.UIAxes_2);
354
355 end
356
357 % Button pushed function: motionbluringButton

```

ii. Hasil Eksekusi





iii. Analisa

Yang dilakukan pada gambar pada prinsipnya adalah High Frequency Boost Filtering(HFBF) yang memperkuat pixel - pixel dengan frequensi tinggi tanpa mengubah pixel - pixel dengan frekuensi rendah

e. Penghilangan Derau Salt and Pepper

i. Kode Program

```
507 function spek5()
508     % Baca citra grayscale
509     gray_image = imread('image-021.bmp');
510     gray_myImage = imread('myImage.bmp');
511
512
513     % Baca citra berwarna
514     color_image = imread('image-022.jpg');
515     color_myImage = imread('myImage.jpg');
516
517     kernelSize = 3
518     color_image_saltpepper = imnoise( color_image , 'salt & pepper' )
519
520     min_denoised_image = min_filter(color_image_saltpepper,kernelSize);
521     max_denoised_image = max_filter(color_image_saltpepper,kernelSize);
522     median_denoised_image = median_filter(color_image_saltpepper,kernelSize);
523     arithmetic_mean_denoised_image = arithmetic_mean_filter(color_image_saltpepper,kernelSize);
524     geometric_denoised_image = geometric_filter(color_image_saltpepper,kernelSize,45);
525     harmonic_mean_denoised_image = harmonic_mean_filter(color_image_saltpepper,kernelSize);
526     % contraharmonic_mean_denoised_image = contraharmonic_mean_filter(color_image_saltpepper,kernelSize, 1.5);
527     midpoint_denoised_image = midpoint_filter([color_image_saltpepper,kernelSize]); You, 35 minutes ago * spek 5 6
528     alpha_trimmed_mean_denoised_image = alpha_trimmed_mean_filter([color_image_saltpepper,kernelSize,2]);
529
530     figure;
531     subplot(5, 2, 1);
532     imshow(gray_image);
533     title('Original Asli');
534
535     subplot(5, 2, 2);
536     imshow(color_image_saltpepper);
537     title('Citra saltpepper');
538
539     subplot(5, 2, 3);
540     imshow(uint8(min_denoised_image));
541     title('Citra min_denoised_image');
542
543     subplot(5, 2, 4);
544     imshow(uint8(max_denoised_image));
545     title('Citra max_denoised_image');
546
547     subplot(5, 2, 5);
548     imshow(uint8(median_denoised_image));
549     title('Citra median_denoised_image');
550
```

```
258 function outputImage = min_filter(inputImage, kernelSize)
259     % dimensi gambar
260     [rows, cols, channels] = size(inputImage);
261
262     % init output
263     outputImage = zeros(rows, cols, channels);
264
265     % radius kernel
266     radius = (kernelSize - 1) / 2;
267
268     for c = 1:channels
269         for i = 1:rows
270             for j = 1:cols
271                 neighborhood = inputImage(max(1, i - radius):min(rows, i + radius), ...
272                                         max(1, j - radius):min(cols, j + radius), c);
273
274                 outputImage(i, j, c) = min(neighborhood(:));
275             end
276         end
277     end
278 end
279
```

```
280 √ function outputImage = max_filter(inputImage, kernelSize)
281     % dimensi gambar
282     [rows, cols, channels] = size(inputImage);
283
284     % init output
285     outputImage = zeros(rows, cols, channels);
286
287     % radius kernel
288     radius = (kernelSize - 1) / 2;
289
290     for c = 1:channels
291         for i = 1:rows
292             for j = 1:cols
293                 neighborhood = inputImage(max(1, i - radius):min(rows, i + radius), ...
294                                         max(1, j - radius):min(cols, j + radius),c);
295
296                 outputImage(i, j,c) = max(neighborhood(:));
297             end
298         end
299     end
300 end
```

```
300 √
301 √ function outputImage = median_filter(inputImage, kernelSize)
302
303     % dimensi gambar
304     [rows, cols, channels] = size(inputImage);
305
306     % init output
307     outputImage = zeros(rows, cols, channels);
308
309     % radius kernel
310     radius = (kernelSize - 1) / 2;
311
312     for c = 1:channels
313         for i = 1:rows
314             for j = 1:cols
315                 % Get the neighborhood of the current pixel
316                 neighborhood = inputImage(max(1, i - radius):min(rows, i + radius), ...
317                                         max(1, j - radius):min(cols, j + radius),c);
318
319                 % Apply the median filter operation
320                 outputImage(i, j,c) = median(neighborhood(:));
321             end
322         end
323     end
324 end
```

```

325
326 ✓ function outputImage = arithmetic_mean_filter(inputImage, kernelSize)
327     % dimensi gambar
328     [rows, cols, channels] = size(inputImage);
329
330     % init output
331     outputImage = zeros(rows, cols, channels);
332
333     % radius kernel
334     radius = (kernelSize - 1) / 2;
335
336     for c = 1:channels
337         for i = 1:rows
338             for j = 1:cols
339                 neighborhood = inputImage(max(1, i - radius):min(rows, i + radius), ...
340                                         max(1, j - radius):min(cols, j + radius),c);
341
342                 outputImage(i, j,c) = mean(neighborhood(:));
343             end
344         end
345     end
346 end
347

```

```

348
349 ✓ function outputImage = geometric_filter(inputImage, kernelSize, angle_degrees)
350
351     % dimensi gambar
352     [rows, cols, channels] = size(inputImage);
353
354     % init output
355     outputImage = zeros(rows, cols, channels);
356
357     % radius kernel
358     radius = (kernelSize - 1) / 2;
359
360     % Convert the angle from degrees to radians
361     angle_radians = deg2rad(angle_degrees); You, 39 minutes ago • spek 5 6
362
363     for c = 1:channels
364         for i = 1:rows
365             for j = 1:cols
366
367                 neighborhood = inputImage(max(1, i - radius):min(rows, i + radius), ...
368                                         max(1, j - radius):min(cols, j + radius),c);
369
370
371                 rotated_neighborhood = imrotate(neighborhood, angle_degrees, 'bilinear', 'crop');
372
373                 center = floor(size(rotated_neighborhood) / 2) + 1;
374
375                 outputImage(i, j,c) = rotated_neighborhood(center(1), center(2));
376             end
377         end
378     end
379 end

```

```
379 end
380 %> function outputImage = harmonic_mean_filter(inputImage, kernelSize)
381
382
383     % dimensi gambar
384     [rows, cols, channels] = size(inputImage);
385
386     % init output
387     outputImage = zeros(rows, cols, channels);
388
389     % radius kernel
390     radius = (kernelSize - 1) / 2;
391
392     for c = 1:channels
393         for i = 1:rows
394             for j = 1:cols
395                 % Get the neighborhood of the current pixel
396                 neighborhood = inputImage(max(1, i - radius):min(rows, i + radius), ...
397                                         max(1, j - radius):min(cols, j + radius),c);
398
399                 reciprocal_neighborhood = 1 ./ double(neighborhood);
400                 harmonic_mean = kernelSize^2 / sum(reciprocal_neighborhood(:));
401
402                 outputImage(i, j,c) = harmonic_mean;
403             end
404         end
405     end
406
407 %> function outputImage = contraharmonic_mean_filter(inputImage, kernelSize, q)
```

```
407 function outputImage = contraharmonic_mean_filter(inputImage, kernelSize, q)
408
409 % dimensi gambar
410 [rows, cols, channels] = size(inputImage);
411
412 % init output
413 outputImage = zeros(rows, cols, channels);
414
415
416 % radius kernel
417 radius = (kernelSize - 1) / 2;
418 for c = 1:channels
419     for i = 1:rows
420         for j = 1:cols
421             neighborhood = inputImage(max(1, i - radius):min(rows, i + radius), ...
422                                         max(1, j - radius):min(cols, j + radius),c);
423
424             numerator = sum(neighborhood(:).^(q + 1));
425             denominator = sum(neighborhood(:).^q);
426
427             if denominator == 0
428                 outputImage(i, j,c) = 0;
429             else
430                 outputImage(i, j,c) = numerator / denominator;
431             end
432         end
433     end
434 end
435
436 end
```

```
437 function outputImage = midpoint_filter(inputImage, kernelSize)
438
439 % dimensi gambar
440 [rows, cols, channels] = size(inputImage);
441
442 % init output
443 outputImage = zeros(rows, cols, channels);
444
445
446 % radius kernel
447 radius = (kernelSize - 1) / 2;
448 for c = 1:channels
449     for i = 1:rows
450         for j = 1:cols
451             neighborhood = inputImage(max(1, i - radius):min(rows, i + radius), ...
452                                         max(1, j - radius):min(cols, j + radius),c);
453
454             outputImage(i, j,c) = median(neighborhood(:));
455         end
456     end
457 end
458
459 end
```

```
460 function outputImage = alpha_trimmed_mean_filter(inputImage, kernelSize, d)
461
462
463     % dimensi gambar
464     [rows, cols, channels] = size(inputImage);
465
466     % init output
467     outputImage = zeros(rows, cols, channels);
468
469
470     % radius kernel
471     radius = (kernelSize - 1) / 2;
472     for c = 1:channels
473         for i = 1:rows
474             for j = 1:cols
475                 neighborhood = inputImage(max(1, i - radius):min(rows, i + radius), ...
476                                         max(1, j - radius):min(cols, j + radius),c);
477
478                 sorted_neighborhood = sort(neighborhood(:));
479
480                 trimmed_neighborhood = sorted_neighborhood((d/2 + 1):(end - d/2));
481
482                 outputImage(i, j,c) = mean(trimmed_neighborhood);
483             end
484         end
485     end
486
487
488
```

ii. Hasil Eksekusi

low Help

Original Asli



Citra salt&pepper



Citra min_denoised,image



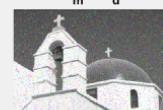
Citra max_denoised,image



Citra median_denoised,image



Citra arithmetic_mean_denoised,image



Citra geometric_denoised,image



Citra harmonic_mean_denoised,image

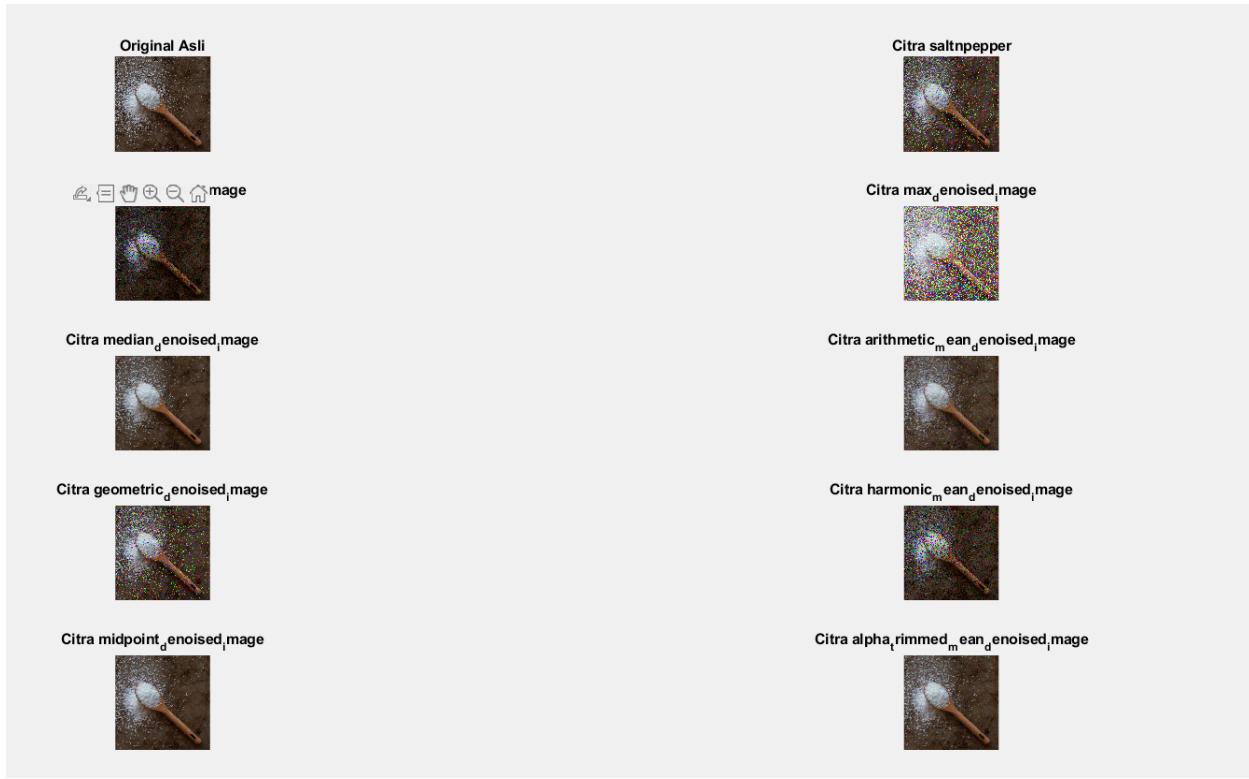


Citra midpoint_denoised,image



Citra alpha_rimmed_mean_denoised,image





iii. Analisa

- Median filter, midpoint, dan alpha rimmed mean berhasil men filter derau salt and paper dengan cukup baik
- Min filter gagal men filter pixel dengan value tinggi
- Max filter gagal men filter pixel dengan value rendah
- Arithmetic Median filter berhasil men filter derau salt and pepper dengan cukup baik
- Geometric filter, dan Harmonic mean gagal memfilter derau salt and pepper

f. Penghilangan Derau Periodik

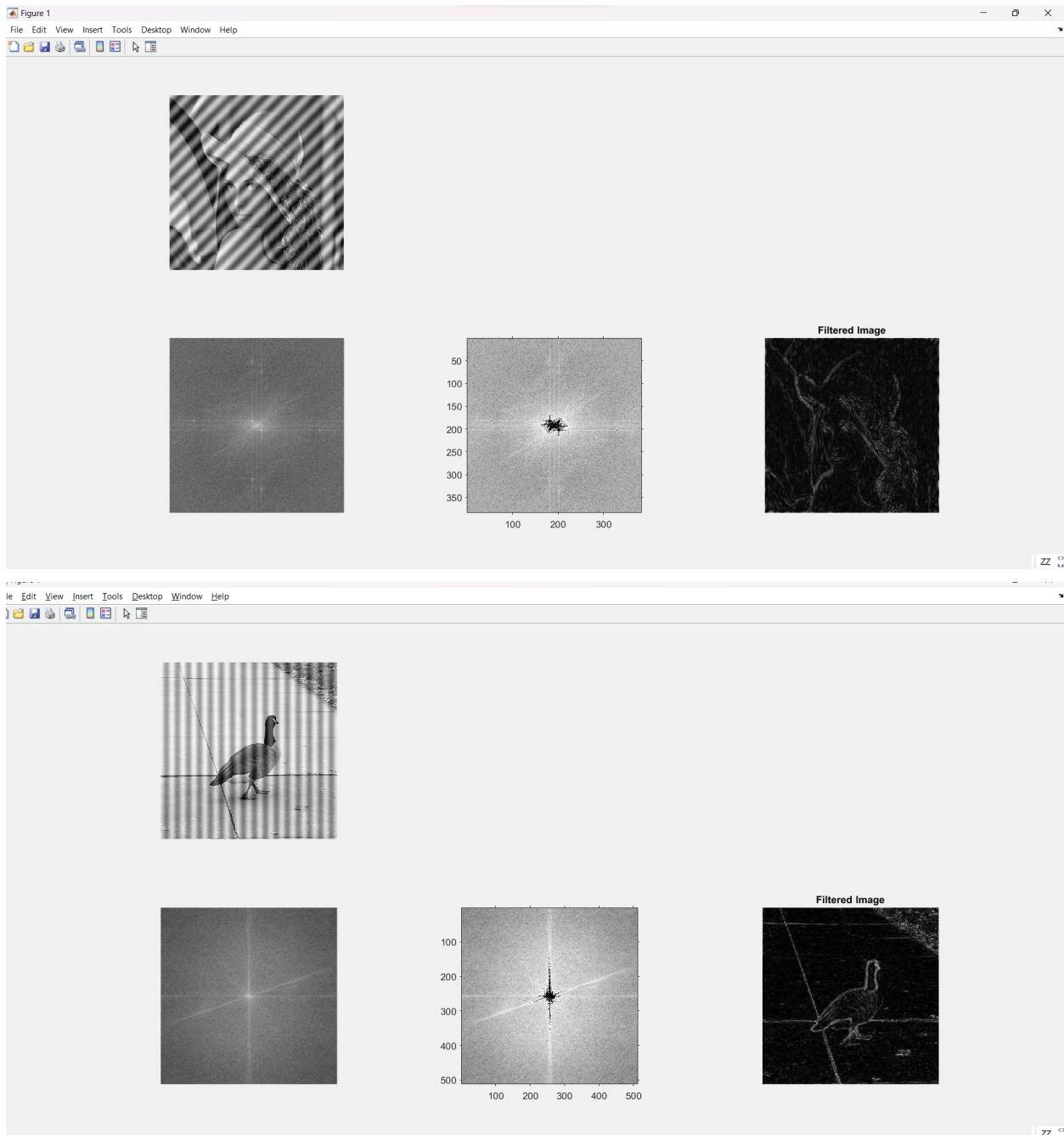
i. Kode Program

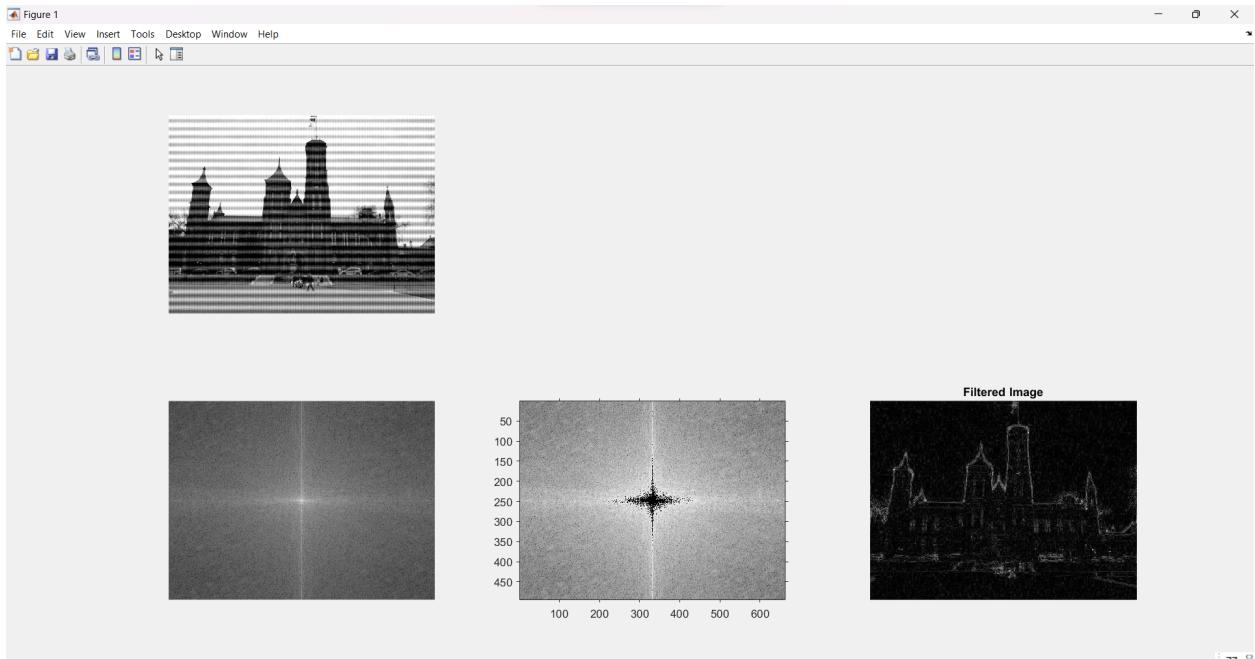
```

555 % Matlab nya.
560 function spek6()
561
562 grayImage = imread('image-029.bmp');
563 [rows columns channel] = size(grayImage);
564 if channel > 1
565 | grayImage = rgb2gray(grayImage);
566 end
567 subplot(2, 3, 1);
568 imshow(grayImage, [0 255]);
569
570 frequencyImage = fftshift(fft2(grayImage));
571 amplitudeImage = log(abs(frequencyImage));
572 minValue = min(min(amplitudeImage))
573 maxValue = max(max(amplitudeImage))
574 subplot(2, 3, 4);
575 imshow(amplitudeImage, []);
576
577 amplitudeThreshold = 10.97;
578 brightSpikes = amplitudeImage > amplitudeThreshold;
579 subplot(2, 3, 5);
580 imshow(brightSpikes);
581
582 brightSpikes(115:130, :) = 0;
583 imshow(brightSpikes);
584 frequencyImage(brightSpikes) = 0;
585 amplitudeImage2 = log(abs(frequencyImage));
586 minValue = min(min(amplitudeImage2))
587 maxValue = max(max(amplitudeImage2))
588 subplot(2, 3, 5);
589 imshow(amplitudeImage2, [minValue maxValue]);
590 axis on;
591
592 filteredImage = ifft2(fftshift(frequencyImage));
593 amplitudeImage3 = abs(filteredImage);
594 minValue = min(min(amplitudeImage3))
595 maxValue = max(max(amplitudeImage3))
596 subplot(2, 3, 6);
597 imshow(amplitudeImage3, [minValue maxValue]);
598 title('Filtered Image');
599
600
601 end

```

ii. Hasil Eksekusi





iii. Analisa

Image masih belum berhasil di filter dengan baik, dikarenakan oleh parameter tuning dari filter masih buruk.

g. Dekonvolusi Motion Blurring

i. Kode Program

```

358 % Button pusher function: motionblurringButtonPushed
359 function motionblurringButtonPushed(app, event)
360     inputImage = imread(app.FileInputnameEditField.Value)
361     kernelSize = 20;
362     angleDegrees = 45;
363
364     motionBlurKernel = fspecial('motion', kernelSize, angleDegrees);
365
366     blurredImage = imfilter(inputImage, motionBlurKernel, 'conv', 'circular');
367
368     % Restorasi citra dengan penapis Wiener
369     wnr1 = deconvwnr(blurredImage, motionBlurKernel, 0); %NSR = 0, tidak ada noise aditif
370
371     title('Citra hasil restorasi');
372     imshow(blurredImage, 'Parent', app.UIAxes);
373
374     imshow(wnr1, 'Parent', app.UIAxes_2);
375
376 end

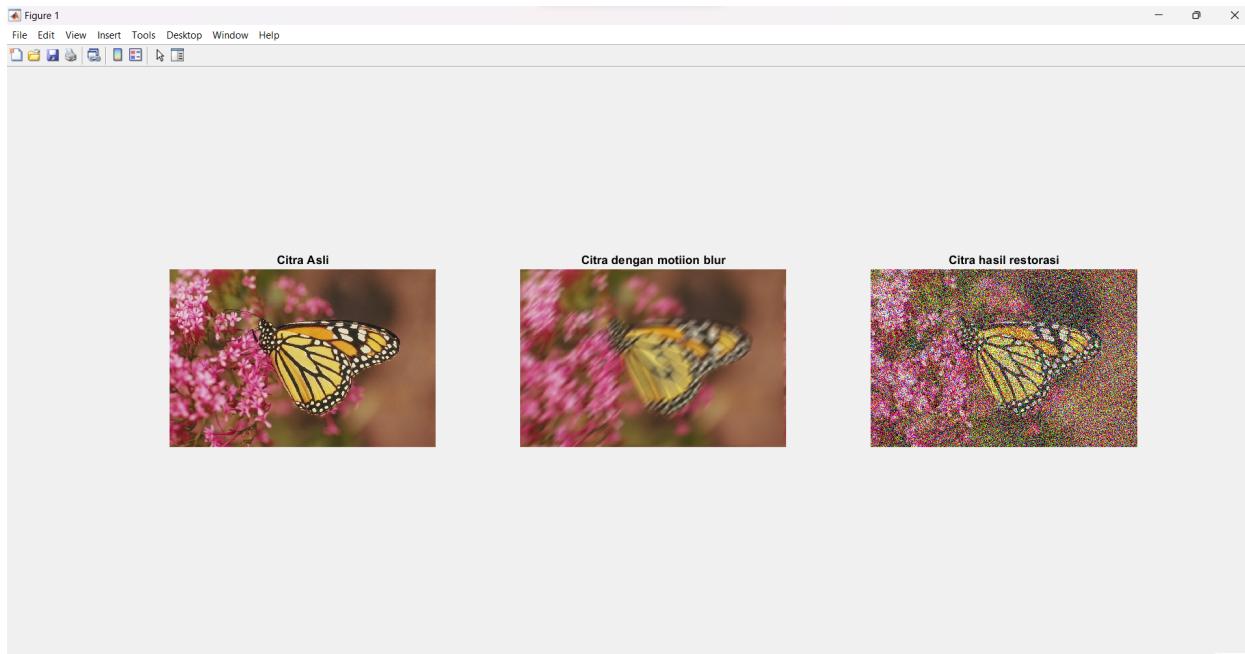
```

```

607
608 |> function blurredImage = motionBlur(inputImage, kernelSize, angleDegrees)
609 |
610 |     % Create a motion blur kernel
611 |     motionBlurKernel = fspecial('motion', kernelSize, angleDegrees);
612 |
613 |     % Apply motion blur to the input image using convolution
614 |     blurredImage = imfilter(inputImage, motionBlurKernel, 'conv', 'replicate');
615 |
616 end
617
618 < function spek7()
619 |
620 |     grayImage = imread('image-032.bmp');
621 |
622 |     colorImage = imread('image-033.jpg')
623 |
624 |     inputImage = colorImage
625 |     % Define the motion blur kernel size and direction
626 |     kernelSize = 30; % Adjust the size as needed
627 |     angleDegrees = 90; % Adjust the angle of motion blur      You, 1 second ago • Uncommitt
628 |
629 |     % Apply motion blur to the input image using convolution
630 |     blurredImage = motionBlur(inputImage, kernelSize, angleDegrees)
631 |
632 |     % Display the original and blurred images
633 |     subplot(1, 2, 1);
634 |     imshow(inputImage);
635 |     title('Citra Asli');
636 |
637 |     subplot(1, 2, 2);
638 |     imshow(blurredImage);
639 |     title('Citra dengan motion blur');
640 |
641 |
642 |     imwrite(blurredImage, "tempBlurredImage.jpg");
643 |
644 end

```

ii. Hasil Eksekusi



MATLAB App

Input Image

Output Image

File Input name: image-033.jpg

Conv

Conv with mask1

$$\frac{1}{16} \times \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Conv with mask2

Conv

Conv with mask3

$\frac{1}{36} \times \begin{bmatrix} 1 & 1 & 2 & 2 & 2 & 1 & 1 \\ 1 & 2 & 3 & 4 & 2 & 1 & 1 \\ 2 & 3 & 4 & 5 & 4 & 2 & 1 \\ 2 & 4 & 8 & 10 & 8 & 4 & 2 \\ 2 & 5 & 10 & 12 & 10 & 5 & 2 \\ 1 & 2 & 3 & 4 & 2 & 1 & 1 \\ 1 & 1 & 2 & 2 & 2 & 1 & 1 \end{bmatrix}$

Conv with mask4

Smoothing

Mean Mask N 5

Gauss Mask Sigma 2

LowPassFilter

ILPF

GLPF

BLPF

HighPassFilter

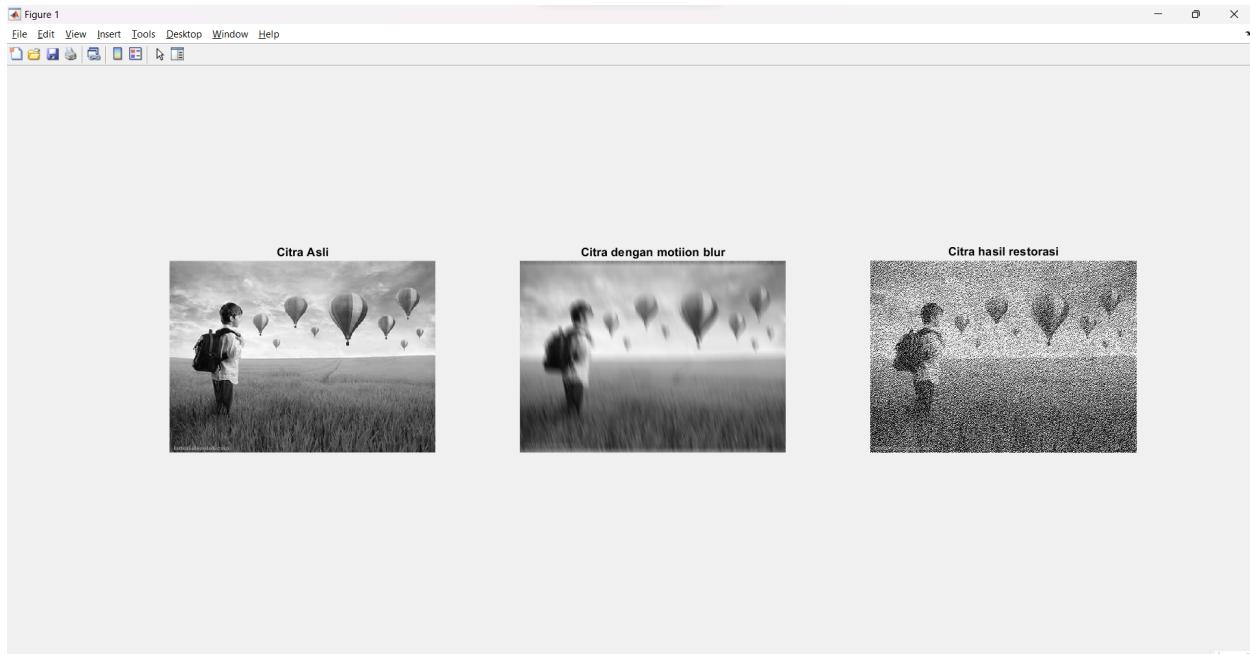
IHPF

GHPF

BHPF

HighBoost

motion bluring



MATLAB App

Input Image

Output Image

File Input name: image-032.bmp

Conv

Conv with mask1 $\frac{1}{4} \times$

1	2	1
2	4	2
1	2	1

Conv with mask2

Conv

Conv with mask3 $\frac{1}{16} \times$

1	1	2	2	2	1	1
1	2	3	3	2	2	1
2	3	4	4	3	2	1
2	4	8	10	4	2	1
1	2	3	3	2	2	1
1	2	2	2	2	1	1
1	1	2	2	2	1	1

Conv with mask4

Smoothing

Mean Mask N 5

0	-1	0
-1	4	-1
0	-1	0

Gauss Mask Sigma 2

-1	-1	-1
-1	17	-1
-1	-1	-1

LowPassFilter

ILPF

HighPassFilter

IHPF

GLPF

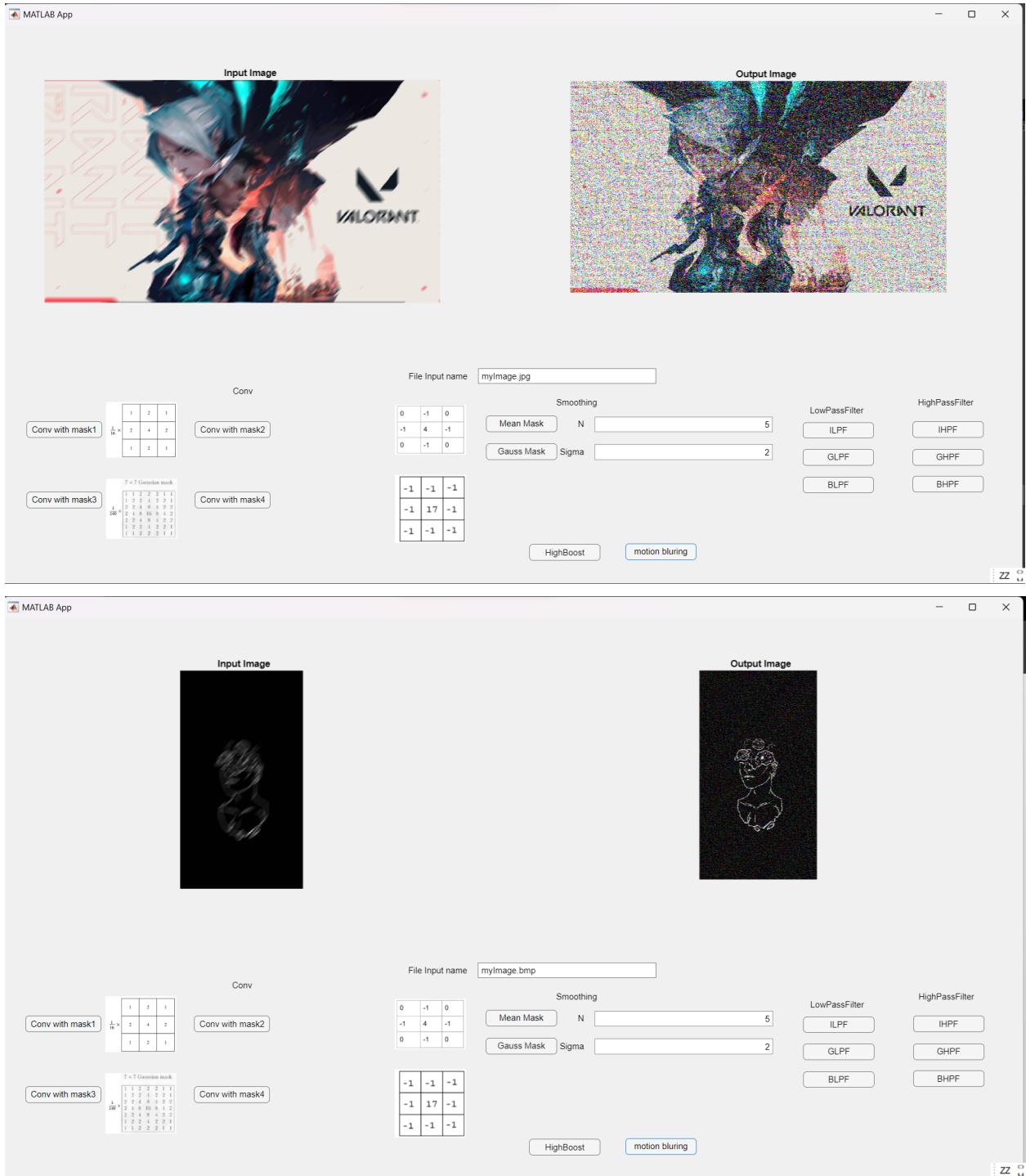
GHPF

BLPF

BHPF

HighBoost

motion bluring



iii. Analisa

Algoritma berhasil menghilangkan motion blurring walaupun menghasilkan bentuk derau lainnya.

3. Alamat Github

<https://github.com/ronggurmahendra/Tugas2-Citra-2023.git>