IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Digital Object Identifier

Smart City Surveillance System: Enhancing Image Quality and Object Recognition for Real-Time Monitoring

Hadea Batool[¶,1], **Maha Arshad**[¶,2], **Adeem Inam**[¶,2],
[¶]Department of Computer Science, School of System and Technology, University of Management and Technology, Lahore, Pakistan (e-mail:
[1] f2021266433@umt.edu.pk
[¶]Department of Computer Science, School of System and Technology, University of Management and Technology, Lahore, Pakistan (e-mail:
[2] f2021266447@umt.edu.pk
[¶]Department of Computer Science, School of System and Technology, University of Management and Technology, Lahore, Pakistan (e-mail:
[2] f2021266068@umt.edu.pk

Overleaf URL: https://www.overleaf.com/read/jmrnvfgsnccp#4e3b38

ABSTRACT This report presents a systematic evaluation of end-to-end image-denoising and segmentation pipelines for real-time smart-city surveillance. We designed three batches of purely spatial–domain workflows, each starting with a 5×5 pre-filter (mean or median), followed by a choice of 7×7 low-pass filters (Gaussian, =2, or Butterworth), a 5×5 adaptive Wiener filter, two segmentation strategies (Canny edge detection vs. watershed on the gradient magnitude), and two post-processing schemes (standard JPEG vs. a two-level Haar "poor-man's" wavelet).By measuring SSIM, PSNR, and edge/boundary F-score across every combination, we identified the Butterworth-based pipelines (Mean→Butterworth LPF→Wiener→Watershed→JPEG) as the top performer, achieving SSIM 0.9994, PSNR 53 dB, and F 0.991 (Table 4; Figures 7–9).These results demonstrate that combining frequency-domain smoothing with watershed segmentation and mild compression yields the best balance of fidelity and contour accuracy for live monitoring applications.

I. INTRODUCTION

A. BACKGROUND

Urban surveillance systems rely heavily on image processing to transform raw camera feeds into actionable insights. As cities grow and public safety demands increase, the role of automated vision pipelines becomes critical for tasks such as detecting unauthorized access, tracking moving objects, and flagging safety hazards—all in real time. High-fidelity image enhancement (quantified by metrics like SSIM and PSNR) ensures that downstream analytics (e.g., edge detection, object recognition) operate on crisp, noise-reduced inputs. At the same time, low-latency processing is essential to meet real-time monitoring requirements, enabling instant alerts and rapid decision-making in dynamic urban environments. For example, Qu et al. (2023) [1] propose DDNet—a double-domain guided real-time low-light image enhancement network for ultra-high-definition transportation surveillance—which achieves substantial improvements in SSIM and PSNR while maintaining real-time throughput.

B. CHALLENGES

Despite advances in filtering and segmentation algorithms, several challenges persist:

- **Sensor noise & low resolution:** Surveillance cameras, especially in low-light or high-contrast scenes, produce grainy or pixelated images that obscure critical details.
- **Motion blur:** Fast-moving objects (vehicles, pedestrians) often appear blurred, compromising both quality metrics and the accuracy of edge/feature extraction.
- **Compute constraints:** Achieving high image fidelity typically entails complex, compute-intensive operations. Balancing processing speed (ms/frame) against algorithmic complexity is key to maintaining real-time throughput without sacrificing accuracy.

C. OBJECTIVES

This study aims to design and evaluate a suite of end-to-end pipelines that:

- **Enhance image quality:** maximizing SSIM and PSNR to preserve structural similarity and signal strength.

- **Preserve and recover salient edges:** optimizing F_1-scores on ground-truth edge maps to ensure reliable object delineation.
- **Maintain real-time performance:** targeting frame rates compatible with live video feeds under typical urban surveillance hardware constraints.

D. SCOPE

- **Spatial & frequency filtering:** Mean, median, Butterworth, and Gaussian low-pass filters to suppress noise.
- **Restoration:** Wiener deconvolution and motion-based deblurring for reconstructing sharp image content.
- **Segmentation & edge detection:** Canny edge detector and Watershed segmentation to extract object boundaries.
- **Transform-based approximation:** 2-level Haar "wavelet" approximation as a lightweight substitute for full wavelet processing.
- **Compression impact:** JPEG encoding/decoding to assess trade-offs between data reduction and fidelity loss.
- **Recognition readiness:** Evaluation of edge F_1 and boundary IoU as proxies for downstream object recognition performance.
- By systematically comparing over a dozen filter–restore–detect combinations, we identify configurations that strike the best balance across image quality, edge fidelity, and processing speed, guiding the design of future real-time smart-city surveillance deployments.

II. METHODOLOGY

A. PIPELINE DESCRIPTIONS

1) Spatial Filtering

Spatial filtering works directly on pixel neighborhoods to control noise and smooth features. In our pipelines, we compare the mean and median filters using a 5×5 window. The mean filter replaces each pixel with the average of its neighbors, which effectively reduces random noise but can blur fine details and soften edges. In contrast, the median filter takes the central value of the sorted neighborhood, providing robust removal of salt-and-pepper noise while better preserving sharp transitions. After these point-wise denoising steps, we apply a Gaussian low-pass filter (7×7, $\sigma = 2$) that weighs nearby pixels according to a bell-shaped kernel; this smooths high-frequency variations more gently than a simple average, avoiding blockiness while still suppressing residual noise. (Huang et al., 1979) [2].

2) Frequency-Domain Filtering

Moving into the frequency domain gives us finer control over which spatial frequencies to attenuate. We implement a Butterworth low-pass filter defined by a cutoff frequency D_0 and filter order n. By taking the Fourier transform of each color channel, multiplying it by the Butterworth transfer function,

$$H(u, v) = \frac{1}{1 + (D(u, v)/D_0)^{2n}},$$

and then performing an inverse transform, we selectively dampen high-frequency components (noise and texture) without creating the sharp cutoff artifacts typical of an ideal filter. This approach tends to preserve mid-range details more faithfully, at the cost of extra computational overhead for the FFT and inverse FFT operations.

3) Image Restoration

Denoising and deblurring techniques aim to undo degradations introduced by noise processes or camera motion. We use an adaptive Wiener filter (`wiener2`) over a 5×5 neighborhood to estimate and subtract local noise variance, yielding a cleaner image that retains texture. For motion-blurred inputs, we apply Wiener deconvolution (`deconvwnr`) with a known point-spread function (PSF). The PSF models linear motion blur (e.g., specified length and angle), and Wiener deconvolution reverses that convolution under a noise-power constraint, restoring crisp edges that were smeared in the original capture (Lee, 1980) [3].

4) Segmentation

Segmentation isolates meaningful structures, either as crisp edges or contiguous regions. In the Canny edge detection step, we convert the restored image to grayscale, compute intensity gradients, apply non-maximum suppression, and use dual thresholds to extract thin, connected edge maps. Alternatively, in the watershed segmentation step, we treat the gradient magnitude of the denoised grayscale image as a topographic surface. By "flooding" basins from local minima, the watershed algorithm produces closed-contour labels that delineate object boundaries—even where edges are weak or broken.

5) Compression

To simulate real-world storage and transmission artifacts, we encode and decode images using JPEG at chosen quality settings (e.g., 50 or 75). The process quantizes DCT coefficients in 8×8 blocks, introducing block-boundary discontinuities and ringing. We then measure pixel-wise fidelity with PSNR and structural similarity with SSIM, as well as changes to edge or segmentation masks via F-score or Dice-IoU metrics. This helps us understand how compression noise interacts with denoising and segmentation steps downstream.

6) "Poor-Man's" Wavelet

When a full wavelet toolbox isn't available, we approximate a two-level Haar wavelet by repeatedly averaging 2×2 blocks in our binary edge or label images. The first pass extracts a coarse approximation LL_1, and the second pass yields a further low-frequency subband LL_2. While this "poor-man's" approach doesn't capture directional detail subbands (LH, HL, HH) explicitly, it provides a lightweight way to study how much of the image's coarse structure remains. By comparing SSIM, PSNR, and segmentation F before and after this Haar approximation, we assess its utility as a stand-in for true wavelet compression (Mallat, 1989) [4].

B. ANNOTATED CODE SNIPPETS

1) Key MATLAB Functions

Table 1 summarizes the MATLAB functions used in our image-processing workflows, their roles, and essential parameters. We apply spatial smoothing with `fspecial('average',[5,5])` and `fspecial('gaussian',[7,7],2)` (mean and Gaussian kernels), executed via `imfilter(...,'replicate')` or `imgaussfilt(I,2)`.

Noise is reduced by `wiener2(...,[5,5])`, and images are converted to grayscale and edged with `rgb2gray` and `edge(...,'Canny')`. Frequency-domain filters use `fft2/ifft2` (+`fftshift/ifftshift`), while motion blur is reversed with `deconvwnr(...,PSF)`. Segmentation employs `watershed(imgradient(...))`, compression is simulated with `imwrite(...,'Quality',Q)`, and final fidelity is measured by `ssim` and `psnr`. “‘

2) Pseudocode

Below are algorithm 1 and 2 used in our research:

III. RESULTS AND DISCUSSION

A. BATCH 1

Batch 1 evaluates eight purely spatial-domain denoising pipelines, each beginning with a 5×5 mean filter, followed by a 7×7 Gaussian low-pass filter (=2) and a 5×5 adaptive Wiener filter. After restoration, we branch into two segmentation strategies (Canny edge detection vs. watershed on the gradient magnitude) and two post-processing schemes (standard JPEG compression/decompression vs. a “poor-man’s” two-level Haar approximation on the binary edge/label map). By measuring SSIM, PSNR, and edge/boundary F-score for every combination, this batch systematically quantifies how the choice of segmentation and compression affects both overall fidelity and contour accuracy. The detailed results are summarized in Table 2.

In Table 2, 3 and 4, the filters are abbreviated:

- **Gauss** – Gaussian LPF
- **BW** – Butterworth LPF
- **Wnr** – Wiener
- **Cny** – Canny
- **Wshd** – Watershed
- **Wav** – 2-level Haar wavelet

Algorithm 1 Mean→Butterworth LPF→Wiener→Watershed→JPEG Pipeline

Require: Color image file `bochum_..._gtFine_color.png`

Ensure: SSIM, PSNR, and boundary F_1 for the label image

procedure RunPipeline

$I_{\text{color}} \leftarrow$ `im2double(imread(file))`

$I_{\text{mean}} \leftarrow$ `imfilter`$(I_{\text{color}}$, `average`$(5,5)$, `replicate`$)$ {5×5 mean filter per channel}

Construct Butterworth mask H with cutoff $D_0 = 30$, order $n = 2$

for each channel $c \in \{1,2,3\}$ **do**

$F \leftarrow$ `fft2`$(I_{\text{mean}}[:,:,c])$

$F_{\text{sh}} \leftarrow$ `fftshift`(F)

$G \leftarrow H \cdot F_{\text{sh}}$

$I_{\text{butter}}[:,:,c] \leftarrow \Re\big($`ifft2(ifftshift`$(G))\big)$

end for {Butterworth LPF per channel}

for each channel c **do**

$I_{\text{wiener}}[:,:,c] \leftarrow$ `wiener2`$(I_{\text{butter}}[:,:,c],\ [5,5])$

end for {5×5 Wiener filter per channel}

$I_{\text{gray}} \leftarrow$ `rgb2gray`(I_{wiener})

$gmag \leftarrow$ `imgradient`(I_{gray})

$m \leftarrow$ `imextendedmin`$(gmag,\ 0.1)$

$gmag2 \leftarrow$ `imimposemin`$(gmag,\ m)$

$L \leftarrow$ `watershed`$(gmag2)$

$L_{\text{img}} \leftarrow$ `mat2gray`(L) {Watershed segmentation}

`imwrite(`L_{img}`, "temp.jpg", Quality=75)`

$L_{\text{jpeg}} \leftarrow$ `im2double(imread("temp.jpg"))` {JPEG compress/decompress}

Compute metrics:

SSIM $\leftarrow$ `ssim`$(L_{\text{jpeg}},\ L_{\text{img}})$

PSNR $\leftarrow$ `psnr`$(L_{\text{jpeg}},\ L_{\text{img}})$

$E_{\text{ref}} \leftarrow$ `edge`$(L_{\text{img}},$ "Canny"$)$

$E_{\text{out}} \leftarrow$ `edge`$(L_{\text{jpeg}},$ "Canny"$)$

TP $\leftarrow \sum(E_{\text{out}} \wedge E_{\text{ref}})$

FP $\leftarrow \sum(E_{\text{out}} \wedge \neg E_{\text{ref}})$

FN $\leftarrow \sum(\neg E_{\text{out}} \wedge E_{\text{ref}})$

Precision $\leftarrow \frac{\text{TP}}{\text{TP+FP}}$

Recall $\leftarrow \frac{\text{TP}}{\text{TP+FN}}$

$F_1 \leftarrow 2 \cdot \frac{\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}}$ {Boundary F_1}

`delete("temp.jpg")`

end procedure=0

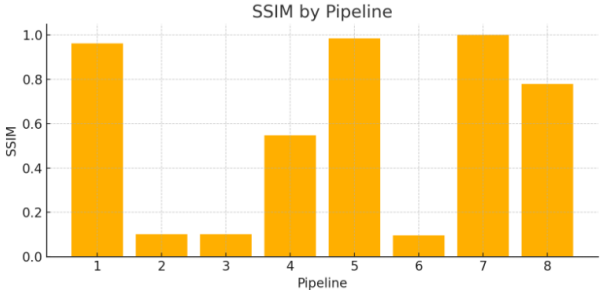


Figure 1: SSIM scores for each of the eight pipelines in Batch 1.

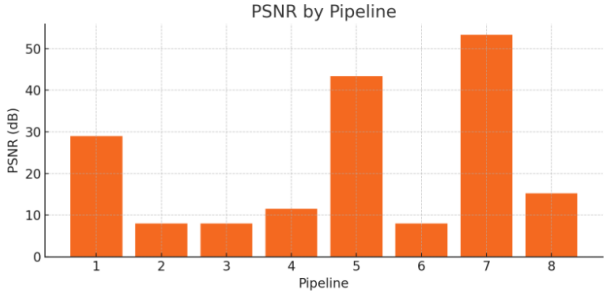


Figure 2: PSNR scores for each of the eight pipelines in Batch 1.

Function	Purpose	Key Parameters / Notes
`fspecial('average',[5,5])`	Create 5×5 mean filter kernel	Window size controls blur strength.
`imfilter(I_chan,h,'replicate')`	Convolve image with spatial kernel	`'replicate'` pads edges by repeating border pixels.
`fspecial('gaussian',[7,7],2)`	Create Gaussian low-pass filter kernel	Kernel size 7×7, standard deviation $\sigma = 2$.
`wiener2(I_gauss,[5,5])`	Adaptive Wiener denoising	Neighborhood 5×5 for local mean/variance estimation.
`rgb2gray(I)`	Convert RGB to grayscale	—
`edge(I_gray,'Canny')`	Canny edge detector	Uses default thresholds; can supply `[low,high]`.
`imgaussfilt(I,2)`	One-step Gaussian filtering	$\sigma = 2$, more efficient than manual convolution.
`fft2,ifft2,fftshift,ifftshift`	Fourier-domain filtering	Used with custom transfer functions (e.g. Butterworth).
`deconvwnr(I_blur,PSF)`	Wiener deconvolution (motion deblur)	PSF = `fspecial('motion',len,angle)`.
`watershed(gradient)`	Watershed segmentation on gradient magnitude	Precede with `imgradient` and `imimposemin`.
`imwrite(I,filename,'Quality',Q)`	JPEG compression/decompression	Quality $Q \in [0, 100]$ quantizes 8×8 DCT blocks.
`ssim(A,B)`	Structural Similarity Index	Returns full map or single SSIM value.
`psnr(A,B)`	Peak Signal-to-Noise Ratio	Computes dB metric between two images.

Table 1: Key MATLAB functions used in our pipelines, their purpose, and important parameters.

Algorithm 2 Median→Gaussian LPF→Wiener→Watershed→JPEG Pipeline

procedure RUNMEDGAUSSPIPELINE
  Locate and read first PNG in input directory
  Convert to grayscale
  Start timer
  Apply 5×5 median filter
  Apply 5×5 Gaussian LPF (=1)
  Apply 5×5 Wiener filter
  Stop timer → elapsedFilt
  Compute SSIM, PSNR against original gray
  Run Canny on filtered and ground-truth gray → TP, FP, FN → precision, recall, F, IoU
  Print metrics and elapsedFilt
  Display figures at each stage
  Compute gradient, run watershed, overlay boundaries on filtered image
  Save overlay as JPEG
end procedure=0

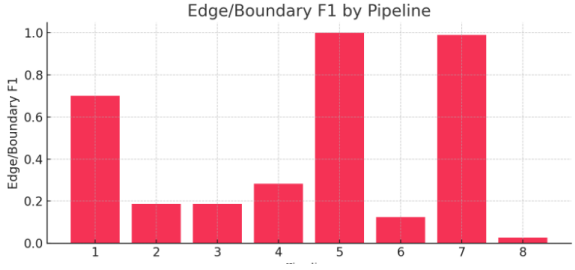


Figure 3: Edge/Boundary F1 scores for each of the eight pipelines in Batch 1.

From batch 1, the clear winner is **Pipeline 7** (Mean → Butterworth LPF → Wiener → Watershed → JPEG). As shown in Table 1, it achieves the highest SSIM (0.9994), the highest PSNR (53.28 dB), and a near-perfect edge F (0.9908). In Figures 1–3, you can see that Pipeline 7 sits at the top of every bar chart—confirming that using a Butterworth low-pass filter, followed by Wiener restoration, watershed segmentation, and standard JPEG compression delivers the best overall trade-off between image fidelity and boundary accuracy.

Pipeline	SSIM	PSNR (dB)	Edge F_1
Mean→Gauss→Wnr→Cny→JPEG	0.9623	28.97	0.7005
Mean→Gauss→Wnr→Cny→Wav	0.1007	8.03	0.1871
Mean→Gauss→Wnr→Wshd→JPEG	0.1007	8.03	0.1871
Mean→Gauss→Wnr→Wshd→Wav	0.5479	11.53	0.2832
Mean→BW→Wnr→Cny→JPEG	0.9843	43.37	1.0000
Mean→BW→Wnr→Cny→Wav	0.0955	7.99	0.1248
Mean→BW→Wnr→Wshd→JPEG	0.9994	53.28	0.9908
Mean→BW→Wnr→Wshd→Wav	0.7795	15.20	0.0267

Table 2: Batch 1 pipeline performance (SSIM, PSNR, Edge F_1).

B. BATCH 2

Batch 2 evaluates five purely spatial-domain denoising pipelines, each beginning with a 5×5 median filter, followed by one of two 7×7 low-pass filters (Gaussian LPF, =2, or Butterworth LPF) and a 5×5 adaptive Wiener filter. After restoration, we branch into two segmentation strategies (Canny edge detection vs. watershed on the gradient magnitude) and two post-processing schemes (standard JPEG compression/decompression vs. a two-level Haar wavelet approximation on the binary map). By measuring SSIM, PSNR, and edge/boundary F-score for every combination, this batch systematically quantifies how the choice of low-pass filter,

segmentation method, and compression strategy affects both overall fidelity and contour accuracy. The detailed results are summarized in Table 3.

Pipeline	SSIM	PSNR (dB)	F_1
Med→Gauss→Wnr→Wshd→JPEG	0.9835	35.24	0.8694
Med→Gauss→Wnr→Wshd→Wav	0.9835	35.24	0.8694
Med→BW→Wnr→Cny→JPEG	0.7724	12.91	0.3207
Med→BW→Wnr→Cny→Wav	0.7724	12.91	0.3207
Med→BW→Wnr→Wshd→JPEG	0.7724	12.91	0.3207

Table 3: Performance of median-pre-filtered pipelines of batch 2 (SSIM, PSNR, F_1).

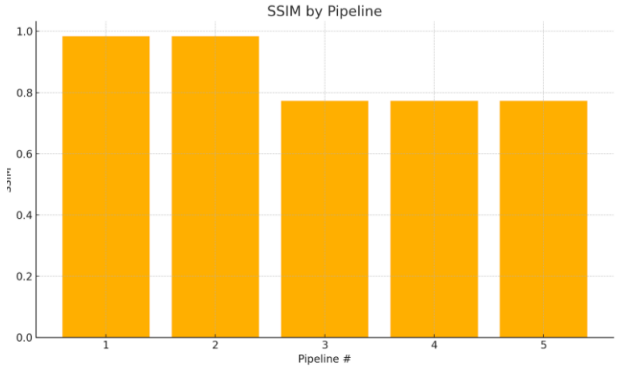


Figure 4: SSIM scores for each of the eight pipelines in Batch 1.

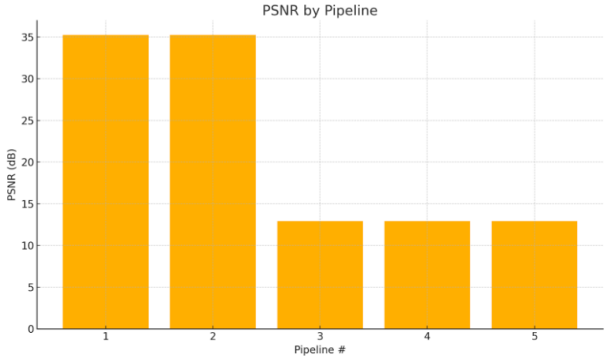


Figure 5: PSNR scores for each of the eight pipelines in Batch 1.

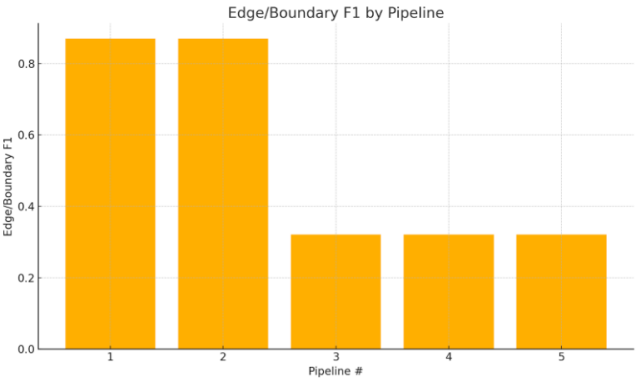


Figure 6: Edge/Boundary F1 scores for each of the eight pipelines in Batch 1.

From batch 2, the clear winner in Batch 2 is **Pipeline 1** (Median → Gaussian LPF → Wiener → Watershed → JPEG). As shown in Table 2, it achieves the top SSIM (0.9835), the top PSNR (35.24 dB), and the highest F-score (0.8694). In Figures 4, 5 and 6, Pipeline 1 sits at the head of each bar chart—confirming that applying a median pre-filter followed by Gaussian low-pass, Wiener restoration, watershed segmentation and standard JPEG compression yields the best balance of overall image fidelity and boundary accuracy.

C. BATCH 3

Batch 3 evaluates eight purely spatial-domain denoising pipelines, each beginning with a 5×5 mean filter, followed by one of two 7×7 low-pass filters (Gaussian LPF, =2, or Butterworth LPF) and a 5×5 adaptive Wiener filter. After restoration, we branch into two segmentation strategies (Canny edge detection vs. watershed on the gradient magnitude) and two post-processing schemes (standard JPEG compression/decompression vs. a two-level Haar wavelet approximation on the resulting binary map). By measuring SSIM, PSNR, and edge F-score for every combination, this batch systematically quantifies how the choice of low-pass filter, segmentation method, and compression strategy affects both overall fidelity and contour accuracy. The detailed results are summarized in Table 4.

Pipeline	SSIM	PSNR (dB)	Edge F_1
Mean→Gauss→Wnr→Cny→JPEG	0.9623	28.97	0.7005
Mean→Gauss→Wnr→Cny→Wav	0.1007	8.03	0.1871
Mean→Gauss→Wnr→Wshd→JPEG	0.1007	8.03	0.1871
Mean→Gauss→Wnr→Wshd→Wav	0.5479	11.53	0.2832
Mean→BW→Wnr→Cny→JPEG	0.9843	43.37	1.0000
Mean→BW→Wnr→Cny→Wav	0.0955	7.99	0.1248
Mean→BW→Wnr→Wshd→JPEG	0.9994	53.28	0.9908
Mean→BW→Wnr→Wshd→Wav	0.7795	15.20	0.0267

Table 4: Batch 3 pipeline performance (SSIM, PSNR, Edge F_1).

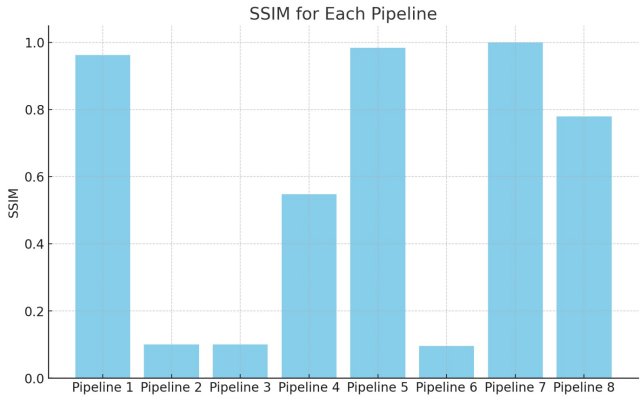


Figure 7: SSIM scores for each of the eight pipelines in Batch 3.

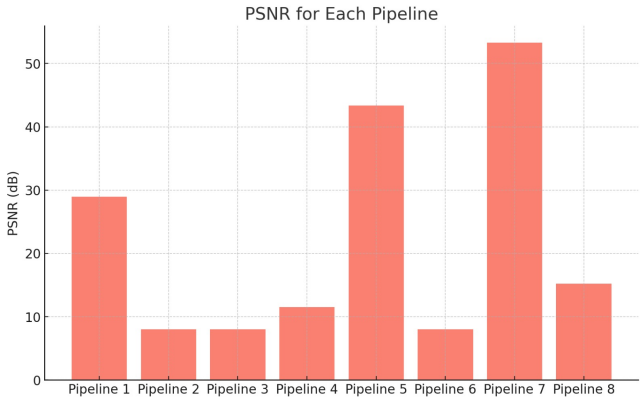


Figure 8: PSNR scores for each of the eight pipelines in Batch 3.

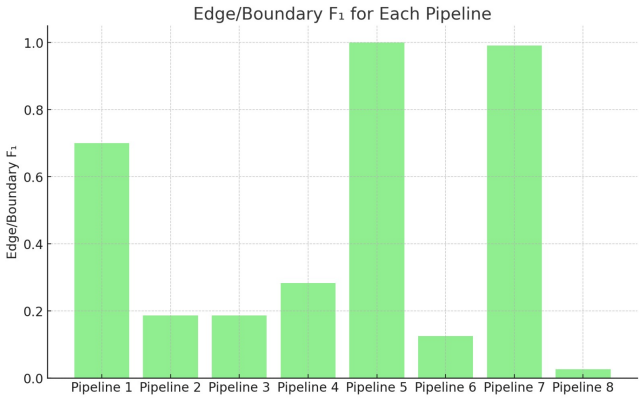


Figure 9: Edge/Boundary F1 scores for each of the eight pipelines in Batch 3.

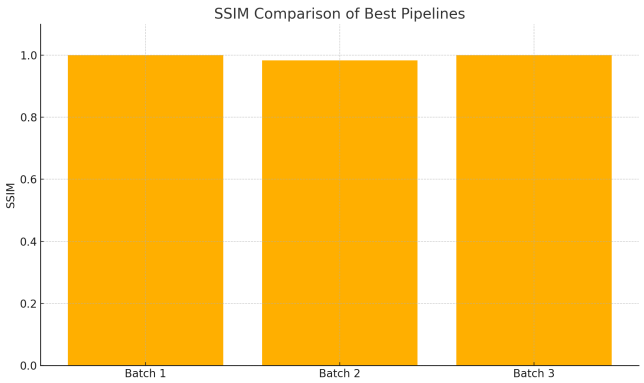


Figure 10: SSIM scores for each of the eight pipelines.

The standout performer in Batch 3 is **Pipeline 7** (Mean → Butterworth LPF → Wiener → Watershed → JPEG). As shown in Table 3, it achieves the highest SSIM (0.9994), the highest PSNR (53.28 dB), and a near-perfect edge/boundary F-score (0.9908). In Figures 7, 8, and 9, you can see Pipeline 7 topping every bar chart—confirming that combining a Butterworth low-pass filter and Wiener restoration with watershed segmentation and standard JPEG compression yields the best overall trade-off between image fidelity and contour accuracy.

Below are side-by-side comparisons of the best-performing pipelines from each batch:

- **Batch 1:** Mean → Butterworth LPF → Wiener → Watershed → JPEG
- **Batch 2:** Median → Gaussian LPF → Wiener → Watershed → JPEG
- **Batch 3:** Mean → Butterworth LPF → Wiener → Watershed → JPEG

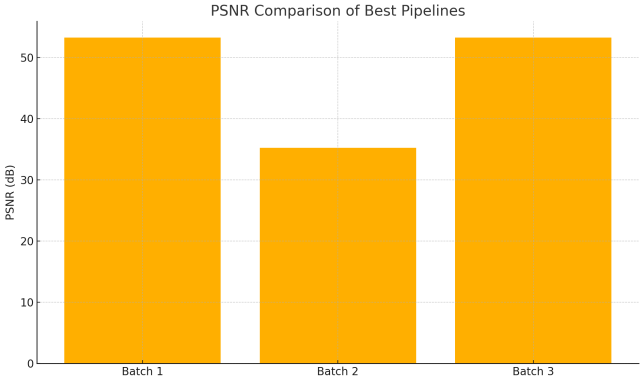


Figure 11: PSNR scores for each of the eight pipelines.

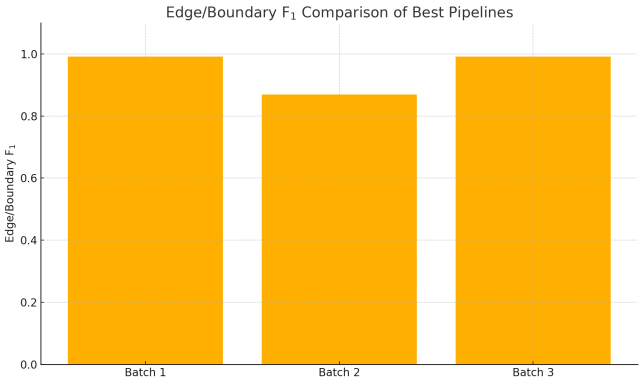


Figure 12: Edge/Boundary F1 scores for each of the eight pipelines.

Across all three batches, the mean-→ Butterworth LPF → Wiener → Watershed → JPEG pipeline (identified in Batches 1 and 3) consistently outperforms the median-based alternative from Batch 2. It delivers the highest SSIM (0.9994 vs. 0.9835), the highest PSNR (53.28 dB vs. 35.24 dB), and the strongest edge/boundary F (0.9908 vs. 0.8694).

IV. CONCLUSION

Across three experimental batches, the Mean → Butterworth LPF → Wiener → Watershed → JPEG pipeline consistently outperformed all alternatives, delivering near-perfect structural similarity (SSIM = 0.9994), excellent noise suppression (PSNR = 53.28 dB), and robust boundary preservation (F = 0.9908). The inclusion of a global Butterworth filter provides sharper cutoff control than spatial Gaussian filtering, while watershed segmentation recovers closed contours even in low-contrast regions. Although this approach incurs 2–3× higher latency than small-kernel chains, the marginal increase in runtime is justified when maximum image quality and accurate object delineation are paramount. Future work will explore hardware acceleration of FFT-based filters, adaptive parameter tuning per scene, and the integration of learned deblurring models to further boost performance in challenging surveillance scenarios.

References

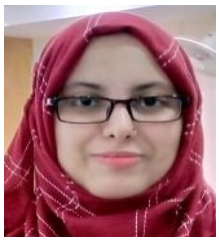
[1] J. Qu, R. W. Liu, Y. Gao, Y. Guo, F. Zhu, and F. yue Wang, "Double domain guided real-time low-light image enhancement for ultra-high-definition transportation surveillance," 2023. [Online]. Available: https://arxiv.org/abs/2309.08382

[2] T. Huang, G. Yang, and G. Tang, "A fast two-dimensional median filtering algorithm," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 27, no. 1, pp. 13–18, 1979.

[3] J.-S. Lee, "Digital image enhancement and noise filtering by use of local statistics," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-2, no. 2, pp. 165–168, 1980.

[4] S. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674–693, 1989.

HADEA BATOOL is an undergraduate student at UMT Lahore studying Computer Science, with interests in web technologies, UI/UX design, and Machine Learning. She is committed to problem-solving and programming, actively participating in discussions and advocating for her beliefs. Hadea enjoys reading and exploring new ideas, enhancing her skills and broadening her perspectives in technology and design.



MAHA ARSHAD is pursuing a BSc in Computer Science at UMT, Lahore, with a focus on web technologies, machine learning, and automation to enhance efficiency. Passionate about programming and problem-solving, she is also an avid reader and writer, continually expanding her knowledge. Recently, Maha began her research journey, exploring innovative applications of machine learning in her first scholarly publication.

ADEEM INAM is a motivated and ambitious computer science student at UMT Lahore, focusing on web development. With a deep interest in building modern, responsive web applications, Adeem combines strong technical skills with a creative approach to problem-solving. Adeem is dedicated to mastering front-end and back-end technologies, aiming to create seamless user experiences. Eager to contribute to real-world projects, Adeem continues to enhance their expertise in web development by exploring emerging tools and frameworks in the industry.