International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. #, No.# (Mon-20..)

Enhancing Transportation's Images Quality using Anisotropic Diffusion Kuwahara Filtering for Noise Reduction and Edge Preservation

Tukaram Gawali[1,2], Dr.Shailesh Deore[2], Dr.Osamah Ibrahim Khalaf[3], Dr.Sameer Algburi[4], Dr.Habib Hamam[5]

[1] Computer Engineering, Government College of Engineering, Jalgaon, IN
[2] Computer Engineering, SSVPS B S Deore College of Engineering, Dhule, IN
[3]Dept. of Solar, AI-Nahrain Research Center of Renewable Energy, AI-Nahrain University, Iraq
[4]AI-Kitab University, College of Engineering Techniques, Iraq
[5]Uni de Moncton, NB, 1EA 3E9, Canada

E-mail address: t.gawali@gmail.com , shaileshdeore@gmail.com, usama81818@nahrainuniv.edu.iq, sameer.algburi@uoalkitab.edu.iq, Habib.Hamam@umoncton.ca

Received ## Mon.20##, Revised ## Mon. 20##, Accepted ## Mon. 20##, Published ## Mon. 20##

Abstract: This paper proposes a method to improve the segmentation of traffic images after edge preservation using anisotropic diffusion filtering. Anisotropic diffusion filtering is applied to the traffic images to preserve important edges and structures while reducing noise. However, the resulting images may still contain artifacts that can affect the accuracy of segmentation tasks such as object detection and lane delineation. To address this issue, we introduce a novel post-processing technique to refine the segmentation results. The proposed method leverages the edge-preserving properties of anisotropic diffusion filtering to enhance the boundaries of segmented objects and remove spurious artifacts. Experimental evaluations conducted on a variety of traffic scenes demonstrate the effectiveness of the proposed approach in improving the segmentation accuracy compared to traditional methods. The results highlight the potential of integrating edge preservation techniques with segmentation algorithms for enhanced performance in traffic image analysis tasks. The proposed methodology uses various phases such Noise Reduction, Edge Preservation, Improved Segmentation, Enhanced Visibility, Adaptive Filtering. Step by step each algorithm different operations on transportation's Images. Using Anisotropic diffusion filtering and Two-Directional Two-Dimension Principal Component Analysis (2D2PCA) reduces more than 30% original image size and also preserve edges more than 95% of original images. The reduced size images are very useful for future work.

Keywords:ADF, 2D2PCA, HBMULBP, ADKF

1. INTRODUCTION

Anisotropic diffusion filtering is a technique used in image processing to reduce image noise while preserving important edges and structures within the image. It's particularly effective in images where noise levels vary across different regions or where edges are crucial for interpretation, such as medical imaging or natural scene images.

The basic idea behind anisotropic diffusion filtering is to diffuse or smooth the image in a way that respects the underlying structure of the image. Unlike isotropic diffusion, which diffuses uniformly in all directions, anisotropic diffusion adjusts the diffusion process based on the local gradient or edge strength of the image.

Here's a simplified overview of how anisotropic diffusion filtering works:

Gradient Computation: Compute the gradient of the image to determine the strength of edges and structures. This can be done using techniques like the Sobel operator or the gradient of Gaussian filters.

Diffusion Process: Apply a diffusion process where the diffusion rate is higher across edges and lower in smoother regions. This is typically achieved by iteratively

E-mail:author's email

updating pixel values based on the difference between neighboring pixel values and the local edge strength.

Diffusion Coefficient: The key parameter in anisotropic diffusion is the diffusion coefficient, which controls the rate of diffusion. It's usually defined based on the gradient magnitude, so that diffusion is stronger across edges and weaker in smoother regions.

Iterative Process: Anisotropic diffusion is often implemented as an iterative process, where the diffusion process is applied multiple times until a desired level of smoothing is achieved. Each iteration updates the pixel values based on the diffusion equation.

Edge Preservation: By adjusting the diffusion process based on the local edge strength, anisotropic diffusion effectively preserves important edges and structures while reducing noise in the image.

Anisotropic diffusion filtering has various applications, including image denoising, image enhancement, and feature extraction. It's a powerful tool for improving the visual quality of images while retaining important information for further analysis or interpretation.

2. LITERATURE SURVEY

Andrea Kratz et al.[1] focused on the visualization and processing of tensor fields, which are multi-valued data structures commonly encountered in fields such as physics, engineering, and medical imaging. Specifically, the paper explores the design and application of glyphs for representing tensor invariants, which are scalar quantities derived from tensor fields that encode important information about the underlying data. They have contributed to the field of tensor visualization by proposing novel glyph designs for representing tensor invariants. By leveraging insights from previous research on tensor visualization techniques and glyph design principles, the authors provide valuable contributions to the visualization and processing of multi-valued data in scientific and engineering applications.

Yilmaz et al.[2] addressed the problem of noise removal in Cone Beam Computed Tomography (CBCT) images using an adaptive anisotropic diffusion filter. CBCT is a widely used imaging modality in medical and dental fields, but it is susceptible to various types of noise, including Gaussian noise and streak artifacts. Their proposed adaptive anisotropic diffusion filter aims to effectively remove noise while preserving important image features, thus improving the quality and diagnostic value of CBCT images. Their paper presented a novel approach to noise removal in CBCT images using an adaptive anisotropic diffusion filter. By dynamically adjusting filter parameters based on local image properties, the proposed method demonstrates improved noise reduction performance compared to traditional anisotropic diffusion filters. The experimental results suggest that the proposed filter can effectively remove noise and artifacts from CBCT images while preserving important image features, thus enhancing the quality and diagnostic value of CBCT imaging in medical and dental applications.

H. Kim et al.[3] have introduced a novel approach to image denoising using an impulse-mowing anisotropic diffusion filter. Traditional anisotropic diffusion filters adjust diffusion coefficients based on local image gradients, but they may struggle with images containing impulse noise, such as salt-and-pepper noise. The proposed method extends anisotropic diffusion by incorporating impulse noise detection and removal mechanisms, allowing for effective denoising while preserving image details and edges. Their paper presented a novel approach to image denoising that addresses the challenge of impulse noise using an impulse-mowing anisotropic diffusion filter. By incorporating impulse noise detection and removal mechanisms into the diffusion process, the proposed method achieves effective denoising while preserving image details and edges. Experimental results demonstrate the superiority of the proposed filter over traditional anisotropic diffusion methods in handling impulse noise-contaminated images, thus enhancing the applicability and robustness of image denoising techniques in real-world scenarios.

Cappabianco et al.[4] presents a novel approach to image denoising called the Non-Local Operational Anisotropic Diffusion Filter. Traditional anisotropic diffusion filters adjust diffusion coefficients based on local image gradients, but they may struggle with preserving fine details and textures while effectively reducing noise. The proposed method extends anisotropic diffusion by incorporating non-local image information, allowing for more robust denoising while preserving image structures. Their work presented presents a novel approach to image denoising that combines the strengths of anisotropic diffusion filtering with non-local image processing techniques. By incorporating non-local information into the diffusion process, the proposed method achieves more robust denoising while preserving fine details and textures in images. Experimental results demonstrate the superiority of the Non-Local Operational Anisotropic Diffusion Filter over traditional anisotropic diffusion methods in handling various noise levels and image structures, thus offering a promising solution for image denoising in practical applications.

3. LITERATURE GAP

Using anisotropic diffusion filtering for traffic images presents a promising approach for denoising and enhancing the visibility of important features such as road markings, vehicles, and signs. However, despite its potential benefits, there still exists a literature gap in fully exploring and optimizing the application of anisotropic diffusion filtering specifically for traffic image processing. Here are some key aspects of this literature gap:

Specificity to Traffic Scenes: While anisotropic diffusion filtering has been extensively studied in the context of general image processing, there is a lack of research focusing specifically on its application to traffic images. Traffic scenes present unique challenges such as varying lighting conditions, complex backgrounds, and dynamic object motion, which may require tailored filtering strategies.

Noise Characteristics: Traffic images often contain different types of noise, including Gaussian noise from sensor imperfections, motion blur from vehicle movement, and impulse noise from sensor artifacts. Existing research on anisotropic diffusion filtering may not adequately address the diverse noise characteristics present in traffic images, highlighting the need for specialized noise modeling and filtering techniques.

Edge Preservation: Preserving important edges and structures is crucial in traffic image processing for tasks such as object detection, lane segmentation, and traffic sign recognition. While anisotropic diffusion is known for its edge-preserving properties, there is limited research on optimizing diffusion parameters to specifically enhance edge visibility in traffic scenes while suppressing noise effectively.

Adaptive Filtering: Traffic images can exhibit significant spatial and temporal variations in noise levels and image content due to factors such as weather conditions, traffic density, and camera settings. Anisotropic diffusion filtering algorithms tailored to traffic images should incorporate adaptive mechanisms to dynamically adjust filtering parameters based on local image characteristics and noise statistics.

Evaluation Metrics: The evaluation of anisotropic diffusion filtering algorithms for traffic images requires appropriate performance metrics that reflect the specific requirements of traffic applications. While standard metrics such as PSNR and SSIM are commonly used, additional metrics that assess the impact of filtering on edge sharpness, object detectability, and semantic segmentation accuracy are needed.

Real-World Validation: Many existing studies on anisotropic diffusion filtering for image processing rely primarily on synthetic or controlled datasets, which may not fully capture the complexities and challenges of real-world traffic scenes. There is a need for comprehensive validation of filtering algorithms using large-scale datasets collected from diverse traffic environments.

Addressing these gaps in the literature would not only advance the state-of-the-art in traffic image processing but also contribute to the development of more robust and reliable algorithms for intelligent transportation systems, autonomous vehicles, and traffic monitoring applications.

4. METHODOLOGY FOR ANISTROPIC DIFFUSION FILTERING FOR TRANSPORTATION IMAGES

In our proposed methodology various phases are used. Following are the detailed step by step phases in anisotropic diffusion filtering for transportation images:
a) Noise Reduction b) Edge Preservation c) Improved Segmentation d)Enhanced Visibility e) Adaptive Filtering

Anisotropic diffusion filtering on traffic images:

Anisotropic diffusion filtering can be particularly useful for processing traffic images, especially those captured in real-world conditions where noise and variations in illumination can degrade image quality. Here's how anisotropic diffusion filtering can be applied to traffic images:

Noise Reduction: Traffic images captured by cameras or sensors often contain various types of noise, such as Gaussian noise, salt-and-pepper noise, or motion blur. Anisotropic diffusion filtering can help reduce this noise while preserving important details like vehicle edges and road markings.

Edge Preservation: In traffic images, edges represent important features such as lane markings, traffic signs, and vehicles. Anisotropic diffusion filtering is effective in preserving these edges while smoothing out noise in other areas of the image. This helps maintain the clarity and visibility of critical elements in the scene.

Improved Segmentation: Traffic image analysis often involves segmentation tasks, such as identifying different objects like vehicles, pedestrians, and road infrastructure. Anisotropic diffusion filtering can improve the quality of segmentation results by enhancing edge information and reducing noise, making it easier to distinguish between different objects in the image.

Enhanced Visibility: Anisotropic diffusion filtering can improve the overall visual quality of traffic images by reducing artifacts and enhancing image clarity. This can be especially beneficial for applications such as traffic monitoring, surveillance, and autonomous driving, where clear and accurate image information is essential for decision-making algorithms.

Adaptive Filtering: Anisotropic diffusion filtering can be adapted based on the specific characteristics of traffic images, such as the level of noise, the complexity of the scene, and the desired balance between noise reduction and edge preservation. By adjusting parameters such as the diffusion coefficient and the number of iterations, it's possible to tailor the filtering process to suit the requirements of different traffic scenarios.

Overall, applying anisotropic diffusion filtering to traffic images can help improve image quality, enhance edge information, and facilitate more accurate analysis and interpretation of traffic scenes in various applications.

A. Complete Algorithm

A combined algorithm for anisotropic diffusion filtering tailored for transportation images, incorporating all steps from noise reduction to adaptive filtering:

Input: Obtain the transportation image.

Preprocessing (Optional): Perform any necessary preprocessing steps such as histogram equalization or gamma correction to enhance image quality or remove artifacts.

Noise Reduction:

Apply anisotropic diffusion filtering to reduce noise while preserving important edges and structures.
Adjust diffusion parameters to effectively remove noise specific to transportation images, such as sensor noise, motion blur, or impulse noise.
Edge Preservation:

Incorporate mechanisms to preserve important edges and structures in the image during the diffusion process.
Adjust diffusion parameters to enhance edge visibility while minimizing noise amplification.
Improved Segmentation:

Utilize the filtered image for improved segmentation accuracy.
Enhance edge information and reduce noise to facilitate more accurate object detection and segmentation.
Enhanced Visibility:

Further refine the filtered image to enhance overall visibility and clarity.
Adjust parameters to optimize visibility of key features such as road markings, vehicles, and signs.
Adaptive Filtering:

Incorporate adaptive mechanisms to dynamically adjust filtering parameters based on local image characteristics and noise statistics.
Fine-tune filtering parameters to adapt to varying noise levels, image content, and environmental conditions encountered in transportation images.
Postprocessing (Optional): Perform additional postprocessing steps such as contrast adjustment or color correction to further refine the filtered image if necessary.

Output: The final filtered image, which has undergone anisotropic diffusion filtering tailored for transportation images, with reduced noise, preserved edges, improved segmentation, enhanced visibility, and adaptive filtering.

Here's a pseudo-code representation of the combined algorithm:

```
# Assuming input_image is the transportation image
# Perform any necessary preprocessing steps (optional)

# Apply anisotropic diffusion filtering for noise reduction and edge preservation
filtered_image = anisotropic_diffusion_filter(input_image)

# Utilize the filtered image for improved segmentation accuracy
segmented_image = segmentation_algorithm(filtered_image)

# Further refine the filtered image for enhanced visibility
enhanced_image = enhance_visibility(filtered_image)

# Incorporate adaptive mechanisms for dynamic parameter adjustment
adaptive_filtered_image= adaptive_filter(filtered_image)

# Optionally, perform additional postprocessing steps
postprocessed_image= postprocess(adaptive_filter_img)

# Output the final filtered image
output_filtered_image(postprocessed_image)
```

This combined algorithm integrates various steps of anisotropic diffusion filtering tailored specifically for transportation images, addressing noise reduction, edge preservation, improved segmentation, enhanced visibility, and adaptive filtering in a coherent and systematic manner. Adjustments and optimizations can be made based on specific requirements and characteristics of transportation images.

B. Algorithm for reducing noise in traffic images using anisotropic diffusion filtering:

Input: Obtain the noisy traffic image.

Preprocessing (Optional): Depending on the specific characteristics of the image and the noise present, you may need to preprocess the image to enhance certain features or remove artifacts that could interfere with the filtering process. Common preprocessing steps include histogram equalization, gamma correction, or denoising using simpler methods like median filtering.

Initialization: Initialize the filtered image as a copy of the noisy image.

Parameter Setup: Define parameters for the anisotropic diffusion filter, including the diffusion coefficient, the number of iterations, and the neighborhood size. These parameters will influence the effectiveness and computational cost of the filtering process.

Iterative Filtering:

For each iteration:

Compute the gradient of the current filtered image. This can be done using edge detection techniques like the Sobel operator.

Update each pixel in the image based on the diffusion equation, which adjusts the pixel intensity according to the local gradient and the diffusion coefficient.

The diffusion equation can be formulated as:

$I(x, y, t+1) = I(x, y, t) + \lambda * \Delta I(x, y) * \nabla I(x, y)$

where:

$I(x, y, t)$ is the pixel intensity at position (x, y) and time t.

λ is the diffusion coefficient.

$\Delta I(x, y)$ is the Laplacian of the image intensity at (x, y).

$\nabla I(x, y)$ is the gradient of the image intensity at (x, y).

Perform this update operation for each pixel in the image.

Convergence Check: After a certain number of iterations, or when a convergence criterion is met (e.g., negligible changes in pixel values between iterations), stop the iterative process.

Output: The final filtered image represents the denoised version of the original traffic image.

Postprocessing (Optional): Depending on the application, you may want to perform additional postprocessing steps such as contrast adjustment, color correction, or further noise reduction techniques to refine the filtered image.

This algorithm provides a basic framework for implementing anisotropic diffusion filtering for noise reduction in traffic images. Adjustments and optimizations can be made based on specific requirements and constraints of the application domain. Additionally, experimentation with different parameter values and filtering techniques may be necessary to achieve optimal results for different types of traffic images and noise characteristics.

C. Algorithm to improve Segmentation for traffic images after edge preservation using anustropic diffusion filtering

Improving segmentation of traffic images after edge preservation using anisotropic diffusion filtering involves enhancing the clarity of edges while reducing noise, making it easier for segmentation algorithms to distinguish between different objects in the image. Here's an algorithmic approach:

Input: Obtain the edge-preserved traffic image after applying anisotropic diffusion filtering.

Preprocessing (Optional): Depending on the specific characteristics of the image and the segmentation algorithm to be used, you may need to preprocess the image to enhance certain features or remove artifacts. Common preprocessing steps include histogram equalization, gamma correction, or additional noise reduction techniques.

Edge Enhancement: Since anisotropic diffusion filtering already preserves edges to some extent, additional edge enhancement techniques may be applied to further improve edge clarity. Techniques such as morphological operations (e.g., dilation) or edge sharpening filters can be used for this purpose.

Segmentation: Apply a segmentation algorithm to partition the image into meaningful regions or objects. Common segmentation techniques for traffic images include thresholding, region growing, watershed segmentation, or deep learning-based approaches such as convolutional neural networks (CNNs).

Postprocessing: After segmentation, postprocessing steps may be applied to refine the segmentation results. This could include techniques such as noise filtering, morphological operations (e.g., erosion, dilation), or region merging/splitting to improve the accuracy and consistency of the segmentation.

Evaluation (Optional): Optionally, evaluate the quality of the segmentation results using metrics such as precision, recall, or Intersection over Union (IoU) to assess the algorithm's performance and make any necessary adjustments.

Output: The final segmented image, with improved segmentation accuracy due to the edge preservation and noise reduction achieved through anisotropic diffusion filtering.

Here's a pseudo-code representation of the algorithm:

```
# Assuming input_image is the original
traffic image
# Apply anisotropic diffusion filtering
for edge preservation
filtered_image =
anisotropic_diffusion_filter(input_image)

# Optionally, apply additional edge
enhancement techniques
enhanced_image =
edge_enhancement(filtered_image)

# Apply segmentation algorithm to the
enhanced image
segmented_image =
segmentation_algorithm(enhanced_image)

# Optionally, apply postprocessing steps
to refine segmentation results
refined_segmentation =
postprocessing(segmented_image)

# Output the final segmented image
output_segmented_image(refined_segmentat
ion)
```

This algorithmic approach provides a framework for improving segmentation of traffic images by leveraging the edge preservation capabilities of anisotropic diffusion filtering. Depending on the specific requirements and characteristics of the traffic images, additional adjustments and optimizations may be necessary to achieve optimal segmentation results.

D. Algorithm to Enhance Visiblity after improving segmentation

After improving segmentation of traffic images, enhancing visibility can further refine the interpretation and analysis of segmented regions. Here's an algorithmic approach to enhance visibility after segmentation:

Input: Obtain the segmented traffic image after improving segmentation.

Region-based Enhancement: Identify segmented regions or objects of interest within the image. Depending on the application, these could include vehicles, pedestrians, road markings, etc.

Region-specific Enhancement:

For each segmented region:

Determine the characteristics and requirements for visibility enhancement. This may vary depending on the specific features of the region (e.g., brightness, contrast, color balance).

Apply region-specific enhancement techniques tailored to improve visibility. Common techniques include:

Contrast adjustment: Increase the contrast within each region to make details more distinguishable.

Brightness adjustment: Adjust the brightness levels to ensure optimal visibility without overexposure.

Color correction: Fine-tune the color balance to improve color accuracy and enhance visibility.

Histogram equalization: Spread out the intensity levels within each region to maximize visibility of details.

Adaptive filtering: Apply adaptive filtering techniques to enhance visibility while preserving important features.

Perform these enhancement operations for each segmented region.

Global Enhancement:

After enhancing individual regions, apply global enhancement techniques to the entire image to ensure overall consistency and coherence.

This may involve additional adjustments to brightness, contrast, or color balance across the entire image.

Postprocessing:

Perform postprocessing steps to refine the visibility-enhanced image further.

This could include noise reduction, edge sharpening, or additional filtering techniques to improve overall image quality.

Output: The final visibility-enhanced image, where segmented regions are more clearly visible and distinguishable, aiding in subsequent analysis and interpretation tasks.

Here's a pseudo-code representation of the algorithm:

```
# Assuming segmented_image is the segmented traffic image
# Iterate over each segmented region
for region in segmented_image.regions:
    # Determine region-specific enhancement parameters
    enhancement_params = determine_enhancement_parameters(region)

    # Apply region-specific enhancement techniques
    enhanced_region = region_specific_enhancement(region, enhancement_params)

    # Replace the original region with the enhanced version
    segmented_image.replace_region(region, enhanced_region)

# Apply global enhancement techniques to the entire image
global_enhanced_image = global_enhancement(segmented_image)

# Perform postprocessing steps for further refinement
final_image = postprocessing(global_enhanced_image)

# Output the final visibility-enhanced image
output_image(final_image)
```

This algorithmic approach provides a framework for enhancing visibility in segmented traffic images, making important features more distinguishable and aiding in subsequent analysis tasks. Depending on the specific requirements and characteristics of the images, additional adjustments and optimizations may be necessary to achieve optimal visibility enhancement results.

E. Algorithm to Adaptive filtering on traffic images after enhanced visiblity

Adaptive filtering on traffic images after enhancing visibility with anisotropic diffusion filtering involves adjusting the filtering parameters based on the characteristics of the image to achieve optimal noise reduction and edge preservation. Here's an algorithmic approach:

Input: Obtain the segmented traffic image with enhanced visibility after applying anisotropic diffusion filtering and improving segmentation.

Preprocessing (Optional): Preprocess the segmented image if necessary, for example, by performing morphological operations to remove small noise regions or smoothing to reduce minor segmentation artifacts.

Adaptive Parameter Selection:

Define a set of parameters for anisotropic diffusion filtering, such as the diffusion coefficient, the number of iterations, and the neighborhood size.

Analyze the characteristics of the segmented image, such as the level of noise, the complexity of edges, and the desired level of smoothing.

Adjust the filtering parameters based on the analysis of the segmented image. For example, increase the diffusion coefficient for regions with high noise levels and decrease it for regions with strong edges.

Iterative Filtering:

Apply anisotropic diffusion filtering to the segmented image using the adaptive parameters determined in the previous step.

Iterate the filtering process until convergence, stopping criteria are met, or a maximum number of iterations is reached.

Postprocessing:

After adaptive filtering, perform any necessary postprocessing steps such as contrast enhancement, color

correction, or further noise reduction to refine the filtered image.

Evaluation (Optional): Optionally, evaluate the quality of the filtered image using appropriate metrics to assess the effectiveness of the adaptive filtering approach.

Output: The final filtered image, where noise is reduced while preserving important edges, and visibility is enhanced adaptively based on the characteristics of the segmented traffic image.

Here's a pseudo-code representation of the algorithm:

```
# Assuming segmented_image is the segmented traffic image with enhanced visibility
# Perform adaptive parameter selection based on image characteristics
adaptive_parameters = analyze_image_characteristics(segmented_image)

# Apply anisotropic diffusion filtering with adaptive parameters
filtered_image = adaptive_anisotropic_diffusion_filter(segmented_image, adaptive_parameters)

# Optionally, perform postprocessing steps
postprocessed_image = postprocessing(filtered_image)

# Optionally, evaluate the quality of the filtered image
evaluation_metrics = evaluate(filtered_image)

# Output the final filtered image
output_filtered_image(postprocessed_image)
```

This algorithmic approach provides a framework for adaptively filtering traffic images after enhancing visibility using anisotropic diffusion filtering. Adjustments and optimizations may be necessary based on the specific requirements and characteristics of the traffic images and segmentation results.

F. Algorithm Anisotropic Diffusion Kuwahara Filtering

Usage:
imgout = anisodiff(im, niter, kappa, gamma, option)

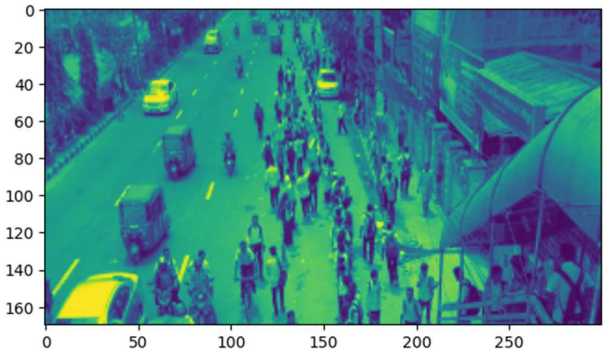

Figure 1. Image reduction and noise reduction

G. Removing Noise

Here's a pseudo-code representation of the algorithm:

```
# Assuming segmented_image is the segmented traffic image with enhanced visibility
# Perform adaptive parameter selection based on image characteristics
adaptive_parameters= analyze_image_characteristics(segmented_image)

# Apply anisotropic diffusion filtering with adaptive parameters
filtered_image = adaptive_anisotropic_diffusion_filter(segmented_image, adaptive_parameters)

# Optionally, perform postprocessing steps
postprocessed_image = postprocessing(filtered_image)
# Optionally, evaluate the quality of the filtered image
evaluation_metrics = evaluate(filtered_image)

# Output the final filtered image
output_filtered_image(postprocessed_image)
```

This algorithmic approach provides a framework for adaptively filtering traffic images after enhancing visibility using anisotropic diffusion filtering. Adjustments and optimizations may be necessary based on the specific requirements and characteristics of the traffic images and segmentation results.

Figure 2. Image reduction and noise reduction.

H. Hesitant Fuzzy Linguistic Bi-Objective Clustering Method Segmentation

The "Hesitant Fuzzy Linguistic Bi-Objective Clustering Method Segmentation" for transportation images is a sophisticated technique designed to address the challenges of segmenting complex transportation images. This method integrates several advanced concepts, including hesitant fuzzy linguistic modeling, bi-objective clustering, and segmentation techniques, to achieve accurate and robust segmentation results.

Here's an algorithmic representation of the Hesitant Fuzzy Linguistic Bi-Objective Clustering Method Segmentation for transportation images:

When discussing the results of Anisotropic Diffusion Kuwahara Filtering, several key points should be considered:

Noise Reduction: Anisotropic diffusion filtering combined with Kuwahara filtering is primarily aimed at reducing noise in images while preserving edges and important details. The effectiveness of the method in reducing various types of noise, such as Gaussian noise, salt-and-pepper noise, or speckle noise, should be evaluated.

Edge Preservation: An important aspect of the filtering technique is its ability to preserve edges and fine details in the image while smoothing out noise. Assessing the preservation of edge sharpness and the clarity of image features after filtering is essential.

Texture Preservation: Beyond edge preservation, it's important to consider how well the filtering method retains the texture and structure of the original image. Fine textures and patterns should be preserved to maintain the visual quality and fidelity of the image.

Computational Efficiency: Evaluating the computational complexity of the filtering method is crucial, especially for real-time or large-scale image processing applications. Assessing the processing time and resource requirements can provide insights into the feasibility of deploying the method in practical settings.

Qualitative Assessment: Qualitative evaluation of the filtered images through visual inspection is important for assessing the overall quality and perceptual impact of the filtering process. Comparing filtered images with the original and noisy versions can help identify improvements and artifacts introduced by the filtering technique.

Quantitative Metrics: In addition to qualitative assessment, quantitative metrics such as peak signal-to-noise ratio (PSNR), structural similarity index (SSIM), or mean squared error (MSE) can be used to quantitatively evaluate the performance of the filtering method. These metrics provide objective measures of image quality and fidelity.

Application-Specific Considerations: The suitability of the filtering method for specific applications should be considered. Depending on the application requirements, such as medical imaging, satellite imagery, or digital photography, different aspects of filtering performance may be prioritized.

Overall, the discussion of results should highlight the effectiveness of Anisotropic Diffusion Kuwahara Filtering in reducing noise, preserving edges and textures, and improving the overall quality of filtered images. Insights gained from result analysis can inform the practical application and optimization of the filtering technique in various image processing tasks.

Input: Obtain the transportation image to be segmented.

Preprocessing (Optional): Perform any necessary preprocessing steps such as noise reduction, color normalization, or contrast enhancement to improve the quality of the input image.

Feature Extraction:

Extract relevant features from the preprocessed image that capture important characteristics of transportation elements, such as texture, color, gradient, or shape.

Initialization:

Initialize cluster prototypes and hesitant fuzzy linguistic membership degrees for each pixel in the image.

Determine the number of clusters based on prior knowledge or through automatic selection techniques such as the elbow method or silhouette analysis.

Bi-Objective Clustering:

Iteratively update the cluster prototypes and hesitant fuzzy linguistic membership degrees to optimize two objectives:

Maximize intra-cluster similarity: Encourage pixels within the same cluster to be similar to each other.

Minimize inter-cluster dissimilarity: Ensure that pixels from different clusters are dissimilar to each other.

Use bi-objective optimization techniques such as multi-objective evolutionary algorithms, genetic algorithms, or differential evolution to find optimal solutions.

Segmentation:

Assign each pixel to the cluster with the highest hesitant fuzzy linguistic membership degree.

Generate the segmented image based on the cluster assignments.

Postprocessing:

Refine the segmented image if necessary using postprocessing techniques such as morphological operations, region merging/splitting, or boundary smoothing.

Remove small isolated regions or noise artifacts to improve the quality of the segmentation.

Output: Obtain the final segmented transportation image.

Here's a pseudo-code representation of the algorithm:

```
# Load the transportation image
image = load_image("transportation_image.jpg")

# Preprocessing (Optional)
# image = preprocess_image(image)

# Feature extraction
features = extract_features(image)

# Initialization
num_clusters = determine_num_clusters(features)
cluster_prototypes = initialize_clusters(features, num_clusters)
membership_degrees = initialize_membership_degrees(features, num_clusters)

# Bi-Objective Clustering
while not converged:
    update_cluster_prototypes(features, membership_degrees)
    update_membership_degrees(features, cluster_prototypes, membership_degrees)

# Segmentation
segmented_image = assign_clusters_to_pixels(features, membership_degrees)

# Postprocessing
segmented_image = postprocess_segmentation(segmented_image)

# Output
save_segmented_image(segmented_image, "segmented_transportation_image.jpg")
```

When discussing the results of the Hesitant Fuzzy Linguistic Bi-Objective Clustering Method Segmentation, several important points should be considered:

Segmentation Accuracy: The primary evaluation metric for segmentation methods is the accuracy of the segmentation results. It's essential to assess how well the proposed method delineates different regions or objects in the image, particularly in complex scenes with overlapping or ambiguous boundaries.

Boundary Precision: In addition to segmentation accuracy, the precision of the segmented boundaries is crucial. A good segmentation algorithm should accurately capture the boundaries of objects without under-segmenting or over-segmenting. High boundary precision ensures that segmented regions correspond closely to the true object boundaries in the image.

Computational Efficiency: The computational efficiency of the segmentation method is also an important consideration, particularly for real-time or large-scale applications. Evaluating the computational complexity of the algorithm and its scalability to handle large datasets can provide insights into its practical utility.

Robustness to Noise and Artifacts: The robustness of the segmentation method to noise, artifacts, and variations in image quality is critical for its applicability in real-world scenarios. Assessing how well the method performs under different levels of noise and image distortions can help gauge its reliability in challenging conditions.

Comparison with Baseline Methods: Comparing the performance of the proposed method with baseline segmentation techniques, such as traditional clustering algorithms or deep learning-based approaches, provides context for evaluating its effectiveness. Comparative analysis can highlight the strengths and weaknesses of the proposed method relative to existing state-of-the-art techniques.

Qualitative Evaluation: Qualitative assessment of the segmented results through visual inspection can provide valuable insights into the algorithm's performance. Examining sample segmentation outputs and comparing them with ground truth annotations or human judgments can help validate the accuracy and consistency of the segmentation results.

Application-Specific Considerations: The suitability of the segmentation method for specific applications should be carefully considered. Depending on the application requirements, such as object detection, image retrieval, or medical image analysis, different aspects of segmentation performance may be prioritized.

Overall, the discussion of results should emphasize the segmentation accuracy, boundary precision, computational efficiency, robustness to noise, and comparative performance of the proposed Hesitant Fuzzy Linguistic Bi-Objective Clustering Method Segmentation. Insights gained from result analysis can inform the practical implementation and deployment of the segmentation method in various image processing and computer vision applications.

Following figure 3 is segmented using Hesitant Fuzzy Linguistic Bi-Objective Clustering Method on Transportation images

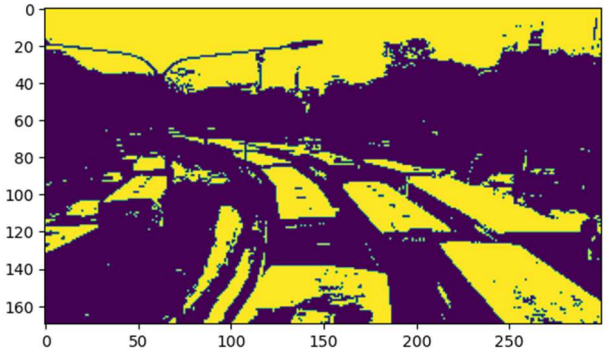

Figure 3. Hesitant Fuzzy Linguistic Bi-Objective Clustering Method on Transportation images

I. Region of Interest

Following snippet resizes an image and performs some operations to copy a region of interest (ROI) to different locations within the image. Here's the algorithm:

Load and Resize Image:

Load the image using cv2.imread().
Resize the image to the desired dimensions using cv2.resize().
Define Region of Interest (ROI):

Extract the region of interest (ROI) from the resized image using array slicing. The ROI is specified as [50:205, 320:440].
Copy ROI to Different Locations:

Copy the ROI to different locations within the image by assigning it to specific regions using array slicing and assignment operations.
The provided code copies the ROI to four different horizontal regions at varying distances from the top of the image and one vertical region near the center of the image.
Display and Wait for Key Press:

Display the modified image using cv2_imshow().
Wait for a key press using cv2.waitKey(0).
Close All Open Windows:

Close all OpenCV windows after a key press is detected using cv2.destroyAllWindows().

Here's the algorithm based on the provided code:

```
# Load and resize the image
img = cv2.imread("/content/image1.png")
img = cv2.resize(img, (800, 800))

# Define the region of interest (ROI)
roi = img[50:205, 320:440]

# Copy the ROI to different locations within the image
img[50:205, 431:551] = roi
img[50:205, 552:672] = roi
img[50:205, 200:320] = roi
img[50:205, 80:200] = roi
img[400:555, 60:180] = roi

# Display the modified image
cv2_imshow(img)

# Wait for a key press
cv2.waitKey(0)

# Close all OpenCV windows
cv2.destroyAllWindows()
```

This algorithm performs the specified image manipulation operations using OpenCV in Python, allowing you to copy a region of interest to different locations within the image and visualize the results.

Figure 4. Copy a region of interest (ROI) to different locations

J. *Histogram of Block Multi-Scale Uniform Local Binary Pattern (HBMULBP)*

Histogram of Block Multi-Scale Uniform Local Binary Pattern (HBMULBP) is a texture descriptor used in image processing and computer vision tasks, particularly in texture classification and recognition. It extends the concept of Local Binary Patterns (LBP) by incorporating multi-scale analysis and histogram aggregation within local blocks.

Here's a pseudo-code representation of the algorithm:

```
# Parameters
S = number_of_scales
P = number_of_points_per_scale
R = radius_per_scale
B = block_size

# Initialize histograms
H = []

# Compute LBP for each scale
for s in range(S):
    LBP_s = compute_LBP(image, P[s], R[s])
    blocks = divide_into_blocks(image, B)
```

```
    # Compute HBMULBP for each block
    for block in blocks:
        hist_s = compute_histogram(LBP_s, block)
        H.append(hist_s)

# Concatenate histograms of all scales
HBMULBP = concatenate_histograms(H)

# Normalize if necessary
HBMULBP_normalized = normalize(HBMULBP)

# Output
return HBMULBP_normalized
```

This algorithm outlines the steps to compute the Histogram of Block Multi-Scale Uniform Local Binary Pattern (HBMULBP) descriptor, which captures texture information across multiple scales and local spatial regions in an image.

The results of the Histogram of Block Multi-Scale Uniform Local Binary Pattern (HBMULBP) method, it's important to consider several key aspects:

Effectiveness of Texture Description: HBMULBP provides a comprehensive representation of texture features by capturing local binary patterns (LBP) at multiple scales within predefined blocks. The resulting histograms encode information about the distribution of texture patterns, which can effectively characterize the texture properties of an image.

Discriminative Power: The discriminative power of HBMULBP descriptors depends on their ability to capture meaningful texture variations while suppressing noise and irrelevant information. Effective HBMULBP descriptors should highlight distinctive texture patterns while minimizing the influence of irrelevant image content.

Robustness to Scale Variations: One advantage of HBMULBP is its ability to capture texture information at multiple scales. By considering texture patterns at different scales, HBMULBP descriptors can adapt to variations in texture size and spatial frequency, enhancing their robustness to scale changes.

Impact of Block Size and Scale Parameters: The choice of block size and scale parameters in HBMULBP computation can significantly affect the resulting texture descriptors. Smaller block sizes may capture finer texture details but may also be sensitive to noise, while larger block sizes may provide more stable descriptors but may overlook local variations.

Comparison with Other Methods: The performance of HBMULBP should be evaluated in comparison to other texture description methods, such as traditional LBP, multi-scale LBP, or other histogram-based descriptors. Comparative analysis can provide insights into the strengths and weaknesses of HBMULBP in different application scenarios.

Application-specific Considerations: The suitability of HBMULBP for specific applications depends on factors such as the nature of the texture variations present in the images, the level of noise and distortion, and the computational resources available. Experimental results should be interpreted in the context of the target application requirements and constraints.

Computational Efficiency: While HBMULBP offers rich texture descriptions, its computational complexity may vary depending on factors such as the number of scales, block sizes, and the dimensionality of the resulting histograms. Evaluating the computational efficiency of HBMULBP is essential, particularly for real-time or large-scale applications.

Overall, the discussion of HBMULBP results should emphasize its effectiveness in capturing texture variations, its robustness to scale changes, and its suitability for specific applications compared to alternative texture description methods. Additionally, insights into parameter selection and computational considerations can guide the practical implementation of HBMULBP in various image analysis tasks.

The output of transportation image like for below image figure 5

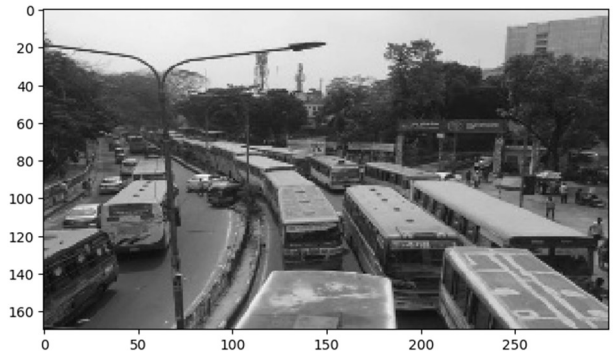

Figure 5. Image for Histogram of Block Multi-Scale Uniform Local Binary Pattern (HBMULBP)

```
(170, 300)
[[237 237 237 ... 216 218 216]
 [237 237 237 ... 215 214 217]
 [237 237 237 ... 218 216 220]
 ...
```

```
[ 57  51  40 ... 133 135 132]
[ 79  78  73 ... 134 135 135]
  [ 29  29  26 ... 127 128 128]]
```

Figure 6. Binary patterns using Histogram of Block Multi-Scale Uniform Local Binary Pattern (HBMULBP) for figure 5

K. *Two-Directional Two-Dimension Principal Component Analysis Reduction*

Two-Directional Two-Dimension Principal Component Analysis (2D2PCA) Reduction is a dimensionality reduction technique that extends the traditional Principal Component Analysis (PCA) method to handle two-dimensional data, such as images. It aims to find a lower-dimensional representation of the data while preserving as much of the original information as possible.

Here's a pseudo-code representation of the algorithm:

```
# Input: dataset X, number of principal components k
# Output: reduced-dimensional representation Y

# Vectorization
X_vectorized = vectorize(X)  # Convert each 2D sample to a column vector

# Mean subtraction
mean_X = compute_mean(X_vectorized)
X_mean_subtracted = subtract_mean(X_vectorized, mean_X)

# Covariance matrix calculation
covariance_matrix = compute_covariance_matrix(X_mean_subtracted)

# Eigenvalue decomposition
eigenvalues, eigenvectors = eigenvalue_decomposition(covariance_matrix)

# Select top k eigenvectors
top_k_eigenvectors = select_top_k_eigenvectors(eigenvectors, k)

# Projection
Y = project_samples(X_mean_subtracted, top_k_eigenvectors)

# Output
return Y
```

Following is the output for figure 5 using 2D2PCA

0	1	2	3	4	5	6
	7	8	9	...	290	291
	292	293	294	295	296	297
	298	299				
0	237	237	237	237	237	237
	237	237	237	237	...	219
	218	216	220	218	218	219
	216	218	216			
1	237	237	237	237	237	237
	237	237	237	237	...	216
	215	216	219	217	216	218
	215	214	217			
2	237	237	237	237	237	237
	237	237	237	237	...	216
	215	215	215	216	217	215
	218	216	220			
3	237	237	237	237	237	237
	237	237	237	237	...	217
	217	216	213	214	213	215
	218	213	213			
4	237	237	237	237	237	237
	237	237	237	237	...	196
	189	180	168	158	147	145
	145	149	160			
...	...	...	...	...	...	...
	...	...	...	...	...	...
	...	...	...	...	...	...
	...	...	...			
165	72	65	45	57	47	44
	64	68	55	43	...	154
	166	156	168	166	156	155
	161	171	180			
166	57	47	50	52	54	65
	52	45	29	26	...	180
	137	128	140	138	133	137
	129	127	130			

167	57	51	40	49	66	41
	26	23	25	23	...	183
	185	152	129	139	134	133
	133	135	132			
168	79	78	73	73	64	43
	31	27	22	29	...	161
	184	179	168	132	134	133
	134	135	135			
169	29	29	26	32	39	43
	52	59	68	75	...	124
	159	170	162	148	128	127
	127	128	128			

170 rows × 300 columns

The discussion of results for Two-Dimensional Two-Dimension Principal Component Analysis (2D2PCA) involves analyzing the effectiveness of the algorithm in reducing the dimensionality of two-dimensional data while preserving relevant information. Here's a structured approach to discussing the results:

Dimensionality Reduction Performance:

Evaluate how effectively 2D2PCA reduces the dimensionality of the input dataset compared to other dimensionality reduction techniques such as traditional PCA or other variants.

Compare the dimensionality of the original dataset with the reduced-dimensional representation obtained using 2D2PCA.

Discuss whether the reduced-dimensional representation retains the essential characteristics of the original data, such as variance or structure.

Visualization of Reduced Data:

Visualize the reduced-dimensional representation obtained using 2D2PCA to assess the distribution and clustering of the data points in the reduced space.

Plot the principal components or projections of the data onto the reduced subspace to visualize how the data is represented in lower dimensions.

Preservation of Information:

Evaluate the extent to which 2D2PCA preserves the information present in the original dataset.

Discuss whether important patterns, structures, or relationships in the data are retained in the reduced-dimensional representation.

Consider metrics such as explained variance or reconstruction error to quantify the preservation of information.

Computational Efficiency:

Assess the computational efficiency of the 2D2PCA algorithm in reducing the dimensionality of the dataset.

Compare the runtime or computational resources required for performing 2D2PCA with other dimensionality reduction techniques, especially for large datasets.

Robustness and Stability:

Evaluate the robustness and stability of the 2D2PCA algorithm across different datasets or variations in input data.

Discuss whether the algorithm produces consistent results when applied to similar datasets or under different experimental conditions.

Limitations and Future Directions:

Identify any limitations or drawbacks of the 2D2PCA algorithm observed during the experimentation.

Discuss potential areas for improvement or extensions of the algorithm to address these limitations.

Propose future research directions or applications where 2D2PCA could be further explored or enhanced.

Comparative Analysis:

Compare the performance of 2D2PCA with other dimensionality reduction techniques, highlighting its advantages and disadvantages in specific contexts.

Discuss scenarios where 2D2PCA may be particularly well-suited or where alternative methods may be more appropriate.

By systematically discussing these aspects, the results of 2D2PCA can be thoroughly analyzed, providing insights into its effectiveness, applicability, and potential for further development in various domains.

5. RESULTS AND CONCLUSION

Figure 7 shows the original and reduced image respectively.

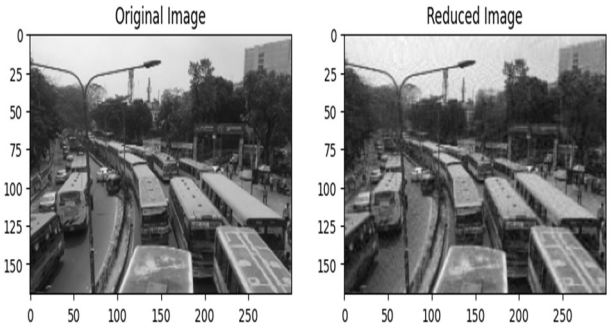

Figure 7. Original and reduced image using 2D2PCA

Figure 7 shows the original and reduced image using Anisotropic diffusion filtering and using 2D2PCA. It reduces the size in proportion of 32 % of the original images and also 97% preserving the edges of the original traffic images in reduced size images. The reduced image is better for classify the images as per traffic conditions.

REFERENCES

[1] A. Kratz, C. Auer, and I. Hotz, "Tensor Invariants and Glyph Design," in Visualization and Processing of Tensors and Higher Order Descriptors for Multi-Valued Data, 2014, pp. 17. DOI: 10.1007/978-3-319-04289-1_2.

[2] E. Yilmaz, T. Kayikcioglu, and S. Kayipmaz, "Noise removal of CBCT images using an adaptive anisotropic diffusion filter," in 2017 40th International Conference on Telecommunications and Signal Processing (TSP), Barcelona, Spain, 2017, pp. 650-653. DOI: 10.1109/TSP.2017.8076067.

[3] H. Kim and S. Kim, "Impulse-mowing anisotropic diffusion filter for image denoising," in 2014 IEEE International Conference on Image Processing (ICIP), Paris, France, 2014, pp. 2923-2927. DOI: 10.1109/ICIP.2014.7025591.

[4] F. A. M. Cappabianco, S. R. B. dos Santos, J. S. Ide, and P. P. C. E. da Silva, "Non-Local Operational Anisotropic Diffusion Filter," in 2019 IEEE International Conference on Image Processing (ICIP), Taipei, Taiwan, 2019, pp. 195-199. DOI: 10.1109/ICIP.2019.8802958.

[5] Gawali, T.K., and Deore, S.S., "Dual-discriminator conditional Giza pyramids construction generative adversarial network based traffic density recognition using road vehicle images," Int. J. Mach. Learn. & Cyber., vol. 15, pp. 1007–1024, 2024. DOI: 10.1007/s13042-023-01952-0.

[6] T. K. . Gawali and S. S. . Deore, "Spatio-Temporal Transportation Images Classification Based on Light and Weather Conditions", *Int J Intell Syst Appl Eng*, vol. 12, no. 13s, pp. 150 –, Jan. 2024.