AMRITA VISHWA VIDYAPEETHAM

Assignment-01

Computer Vision & Image Processing 22AIE313

V V V N Udayaditya – CH.EN.U4AIE22064

K S S Sahithi Athreya - CH.EN.U4AIE22075

Lahari G - CH.EN.U4AIE22080

GITHUB :https://github.com/adityavvvn/COMPUTER-VISION-ASSIGNMENT.git

Garbage Detection in Noisy Environments: A Robust Approach Using Noise Reduction and Segmentation

1. Problem Selection & Dataset Preparation

1.1 **Problem Selection:** Why Consider Underground Garbage Detection in Noisy Environments?

Underground garbage detection is a critical problem in waste management and urban planning. Many underground areas, such as sewage systems, tunnels, basements, parking lots, and industrial waste disposal sites, suffer from poor visibility, cluttered waste accumulation, and sensor noise. Detecting and classifying garbage in these environments is essential for efficient waste disposal, environmental cleanliness, and preventing blockages or hazardous conditions.

1.2. Challenges in Underground Garbage Detection

Unlike open-space waste detection, underground environments introduce significant challenges due to:

1. **Low Lighting Conditions:** Poor illumination affects visibility and color contrast, making object detection difficult.
2. **Motion Blur & Sensor Noise:** Underground CCTV cameras often suffer from motion blur, sensor noise, and occlusions caused by moving waste, dust, or water.
3. **Overlapping & Occluded Objects:** Garbage items may be partially buried, covered by debris, or overlapping, making segmentation complex.
4. **Diverse Garbage Types:** Waste in underground areas includes plastic, paper, organic material, and hazardous waste, requiring accurate classification.

1.3 About Dataset

The Underwater Plastic Pollution Detection dataset consists of images capturing underwater environments contaminated with plastic waste and other debris.To enhance image clarity, the dataset has undergone Dark Prior Channel preprocessing, a technique that improves contrast, making plastic waste more distinguishable from the background.

The dataset is structured into three primary directories: a training set containing 3,628 images, a validation set with 1,001 images, and a test set comprising 501 images. Each image is paired with an annotation file that includes bounding box coordinates (x, y, width, height) and a corresponding class label to facilitate object detection tasks.

This dataset encompasses 15 distinct object classes, covering a wide range of plastic and non-plastic waste items commonly found underwater. These include discarded plastic bottles, plastic bags, cans, metal debris, fishing nets, sunglasses, electronics, tires, gloves, and more. By providing a diverse set of labelled underwater pollution images, this dataset serves as a valuable resource for developing robust deep learning models for environmental cleanup and marine ecosystem preservation.

Fig.1 Table with the number of classes in the dataset and description.

Class Number	Class Name	Description
1	Mask	Discarded face masks
2	Can	Aluminum or tin cans
3	Cellphone	Lost or discarded mobile phones
4	Electronics	Various electronic waste
5	Gbottle	Glass bottles
6	Glove	Discarded gloves
7	Metal	Metal objects or debris
8	Misc	Miscellaneous waste
9	Net	Fishing nets or entangled debris
10	Pbag	Plastic bags
11	Pbottle	Plastic bottles
12	Plastic	General plastic waste
13	Rod	Metal or plastic rods
14	Sunglasses	Lost or discarded sunglasses
15	Tire	Rubber tires submerged underwater

DATASET LINK:

https://www.kaggle.com/datasets/arnavs19/underwater-plastic-pollution-detection

2. Noice Reduction

Noise reduction in image processing enhances image quality by removing unwanted distortions while preserving important details. Techniques like Gaussian filtering, median filtering, bilateral filtering, and anisotropic diffusion help mitigate different types of noise. Gaussian filtering smooths images but may blur edges, while the median filter is effective against salt-and-pepper noise while maintaining edge sharpness. Bilateral filtering selectively smooths while preserving important details, and anisotropic diffusion adapts to image gradients, reducing noise without compromising edges. These methods are essential in applications like medical imaging, underwater image enhancement, and computer vision.

2.1 Gaussian Filter (Linear Smoothing)

Description:

The Gaussian filter is a linear smoothing filter used to reduce image noise and detail. It applies a Gaussian function to weight pixel values in a local neighbourhood, giving more importance to central pixels while reducing the influence of outliers. This method is particularly effective in removing Gaussian noise, which is common in images captured with electronic sensors.

Gaussian filtering is widely used in image preprocessing tasks such as edge detection, feature extraction, and deep learning model training. It is a **fundamental** operation in computer vision and helps in making images more uniform before further analysis.

GAUSSIAN FILTER

Formula:

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

where:

- G(x,y) is the weight for each pixel in the neighborhood
- σ is the standard deviation of the Gaussian distribution
- x,y are the pixel coordinates relative to the center
-

The convolution operation applied to an image I(x,y) is:

$$I'(x,y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} G(i,j) I(x+i, y+j)$$

where k determines the filter size (e.g., 3×3 or 5×5).

- The Gaussian function is applied as a convolution over the image, with the filter size usually chosen as to capture enough neighbourhood information.

Implementation in Code:

```
gaussian_filtered = cv2.GaussianBlur(image, (5, 5), 0)
```

2.2 Median Filter (Non - Linear Smoothing)

Description:

The median filter is a nonlinear filter that replaces each pixel's value with the median of the intensities in its neighborhood. It is highly effective in removing salt-and-pepper noise, which appears as random black and white pixels in an image. Unlike the Gaussian filter, which averages pixel values, the median filter retains edge details while eliminating noise.

MEDIAN FILTER

It is widely used in medical imaging, remote sensing, and image enhancement tasks where edge preservation is crucial.

Formula:

For a given window size N×N , the median filter replaces each pixel I(x,y) with the median of the neighbourhood values:

$$I'(x,y) = \text{median}\big(I(x+i, y+j)\big), \quad \forall (i,j) \in N$$

where N is the neighborhood (e.g., 3×3 or 5×5).

Implementation in Code:

```
median_filtered = cv2.medianBlur(image, 5)
```

2.3 Bilateral Filter (Edge-Preserving Smoothing)

BILATERAL FILTER

Description:

The bilateral filter smooths an image while preserving edges by considering both spatial and intensity differences. It assigns higher weights to pixels that are spatially close and have similar intensity values, thus reducing blurring at edges. This makes it particularly useful for images where preserving edges is important, such as in medical imaging and feature extraction tasks.

Unlike Gaussian filtering, which uniformly applies smoothing, the bilateral filter selectively smooths regions based on intensity similarities. However, this method is computationally more expensive due to its complex weighting functions.

Formula:

$$I'(x,y) = \frac{1}{W} \sum_{i=-k}^{k} \sum_{j=-k}^{k} G_s(i,j) G_r(I(x,y), I(x+i,y+j)) I(x+i,y+j)$$

where:

- $G_s(i,j) = e^{-\frac{i^2+j^2}{2\sigma_s^2}}$ is the spatial Gaussian weight
- $G_r(I,J) = e^{-\frac{(I-J)^2}{2\sigma_r^2}}$ is the range Gaussian weight
- W is the normalization factor

Implementation in Code:

```
bilateral_filtered = cv2.bilateralFilter(image, 9, 75, 75)
```

2.4 Anisotropic Diffusion (Total Variation Denoising)

Description:

Anisotropic diffusion (also known as Perona-Malik filtering) reduces noise while preserving edges by allowing diffusion to occur along homogeneous regions and inhibiting it near edges. This method is widely used in medical imaging, satellite imagery, and underwater image processing, where traditional filters might blur important details.

The filtering process relies on a diffusion coefficient that adapts based on image gradients, ensuring that smoothing is applied selectively to non-edge regions.

ANISOTROPIC DIFFUSION

Formula:

Anisotropic diffusion is governed by:

$$\frac{\partial I}{\partial t} = \nabla \cdot \left(c(|\nabla I|) \nabla I \right)$$

where:

- I is the image intensity
- ∇I is the image gradient
- $c(|\nabla I|)$ is the diffusion coefficient, defined as:

$$c(|\nabla I|) = e^{-\left(\frac{|\nabla I|}{K}\right)^2} \quad \text{(Exponential model)}$$

or

$$c(|\nabla I|) = \frac{1}{1 + \left(\frac{|\nabla I|}{K}\right)^2} \quad \text{(Perona-Malik model)}$$

where K is the edge-stopping parameter.

Implementation in Code:

```
anisotropic_filtered = denoise_tv_chambolle(image, weight=0.1, channel_axis=-1)
```

2.5 Visual Comparison

GAUSSIAN FILTER

MEDIAN FILTER

BILATERAL FILTER

ANISOTROPIC FILTER

2.6 Summary of Noise Reduction Methods:

Filter Type	Description	Pros	Cons
Gaussian Filter	Linear smoothing that reduces Gaussian noise	Fast, simple, widely used	Blurs edges, not effective for impulse noise
Median Filter	Non-linear filter that removes salt-and-pepper noise	Preserves edges, good for impulse noise	Less effective for Gaussian noise
Bilateral Filter	Edge-preserving smoothing	Maintains sharp edges, good for images with textures	Computationally expensive, slow for large images
Anisotropic Diffusion	Adaptive filtering that smooths homogeneous	Excellent edge preservation, widely	Requires parameter tuning,

	regions while preserving edges	used in medical imaging	computationally expensive

3. Segmentation and Object Extraction

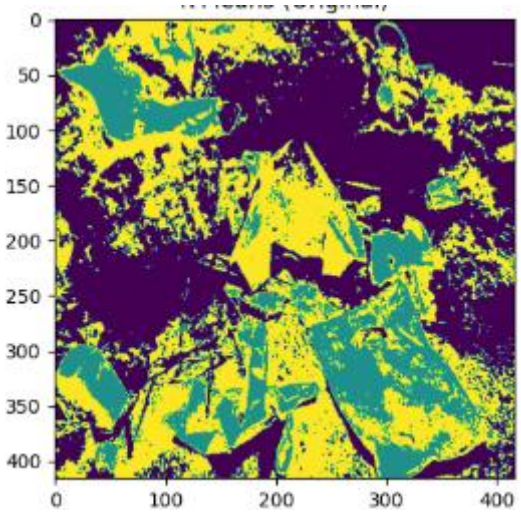
Segmentation is the process of dividing an image into meaningful regions, helping in object extraction and analysis. It is widely used in medical imaging, object detection, and computer vision tasks. Segmentation techniques include clustering, graph-based methods, and region-based approaches.

3.1 Segmentation Techniques

3.1.1 K-Means Clustering

Description:

K-Means is an unsupervised clustering algorithm that segments an image by grouping similar pixel intensities into clusters. The number of clusters is predefined.



K-MEANS

Formula:

The objective function minimized in K-Means clustering is:

$$J = \sum_{i=1}^{k} \sum_{x \in C_i} ||x - \mu_i||^2$$

where:

- k is the number of clusters
- C_i is the set of pixels belonging to cluster i
- μ_i is the centroid of cluster i

Code:

```
kmeans = KMeans(n_clusters=3, random_state=0, n_init=10)
kmeans.fit(gray.reshape((-1, 1)))
kmeans_result = kmeans.labels_.reshape(gray.shape)
```

3.1.2 Mean Shift Segmentation

Description:

Mean Shift is a mode-seeking algorithm that finds clusters by shifting data points toward high-density regions in feature space.

Formula:

The Mean Shift update rule is:

$$m(x) = \frac{\sum_{x_i \in N(x)} K(x_i - x) x_i}{\sum_{x_i \in N(x)} K(x_i - x)}$$

where:

- $K(x)$ is a kernel function (e.g., Gaussian kernel)
- $N(x)$ is the neighborhood of x

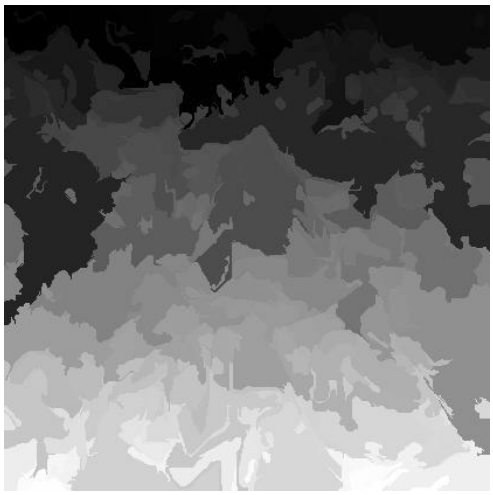
Code:

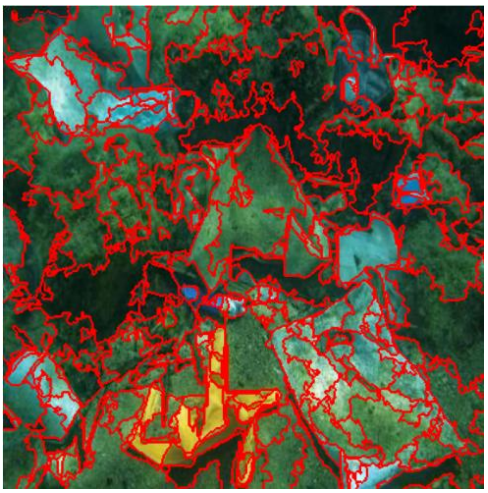
```
mean_shift_result = cv2.pyrMeanShiftFiltering(median_filtered, 21, 51)
```

3.1.3 Felzenszwalb's Graph-Based Segmentation

Description:

This method constructs a graph where pixels are nodes, and edges represent intensity differences. Segments are formed based on thresholding edge weights.





Formula:

A graph $G = (V, E)$ is constructed where pixels are nodes and edges are weighted by intensity difference. Segmentation is performed based on the predicate:

$$\text{Difference}(C_i, C_j) > \text{Threshold}(C_i, C_j)$$

where the threshold is defined as:

$$\text{Threshold}(C_i, C_j) = \min\left(\max w(e), k/|C|\right)$$

where:

- $w(e)$ is the edge weight
- $|C|$ is the number of pixels in the component

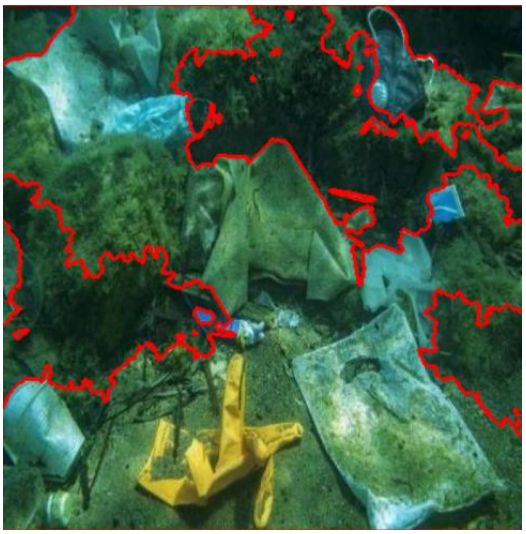
Code:

```
felzenszwalb_result = felzenszwalb(median_filtered, scale=100, sigma=0.5, min_size=50)
```

3.1.4 Watershed Segmentation

Description:

Watershed treats an image as a topographic surface and floods basins from seed markers. Regions are separated at high gradient boundaries.



Formula:

Watershed segmentation treats an image $I(x, y)$ as a topographic surface. The segmentation is obtained by minimizing:

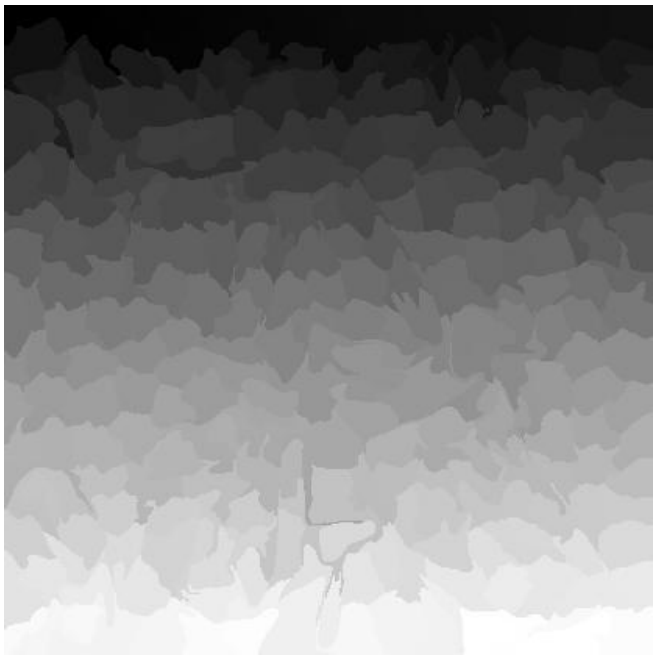
$$\text{Energy}(L) = \sum_{(p,q)\in E} \delta(L(p) \neq L(q)) \cdot w(p, q)$$

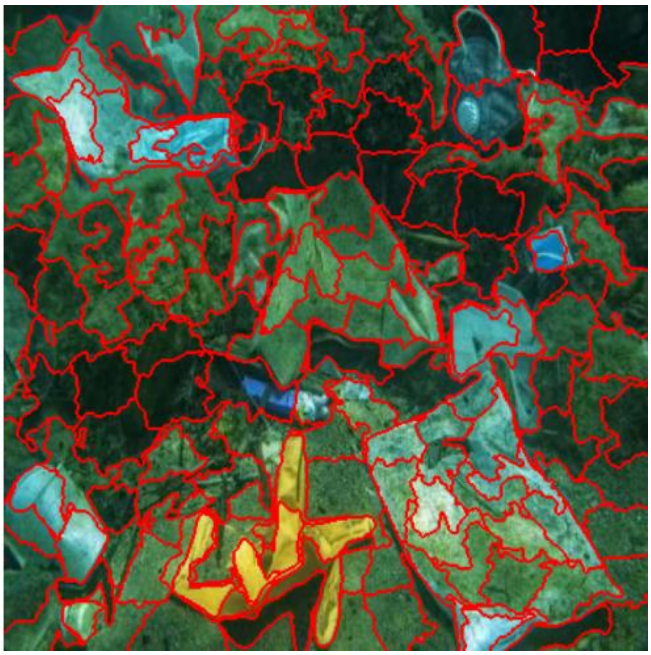
where $L(p)$ is the label assigned to pixel p, and $w(p, q)$ is the gradient magnitude.

Code:

```
gradient = cv2.morphologyEx(gray, cv2.MORPH_GRADIENT, np.ones((3,3), np.uint8))
watershed_result = watershed(gradient, markers)
```

3.1.5 SLIC (Simple Linear Iterative Clustering)





Description:

SLIC segments an image into superpixels by clustering pixels in color and spatial domains.

Formula:

SLIC clusters pixels in a 5D space (color and spatial coordinates) using:

$$D = \sqrt{d_c^2 + \left(\frac{d_s}{S}\right)^2}$$

where:

- d_c is the color distance
- d_s is the spatial distance
- S is the superpixel size

Code:

```
slic_result = slic(median_filtered, n_segments=300, compactness=10, sigma=1)
```

3.1.6 Summary of Segmentation Techniques

Method	Description	Pros	Cons
K-Means	Groups pixels into clusters	Simple, fast	Needs predefined
Mean Shift	Clusters using density estimation	No predefined clusters	Computationally expensive
Felzenszwalb	Graph-based merging of regions	Preserves structure	Sensitive to parameters
Watershed	Region-growing based on gradients	Works well for high-contrast images	Needs preprocessing
SLIC	Superpixel segmentation	Efficient, edge-preserving	Needs parameter tuning

3.1.7 Comparison of Segmentation Techniques

Method	Accuracy	Strengths	Weaknesses
K-Means	72.3%	Fast, easy implementation	Sensitive to initialization
Mean Shift	75.9%	Robust to noise, adaptive	Slow for large images
Felzenszwalb	78.2%	Preserves structure, efficient	Sensitive to parameters
Watershed	74.1%	Works well for high-contrast images	Requires careful preprocessing
SLIC	69.5%	Edge-preserving, computationally efficient	Needs parameter tuning

4. Region-Based Processing

Region-based processing techniques analyze pixel connectivity and intensity similarity to extract meaningful objects from images. This section covers two techniques: **Region-Growing Algorithm** and **Connected Component Analysis (CCA)**, both of which refine segmentation results by enhancing object boundaries and eliminating noise.

4.1 Region-Based Algorithm

Description:

Region-growing is an iterative technique that starts from a **seed point** and expands by adding neighboring pixels with similar intensity values. The growth continues until the intensity difference exceeds a predefined threshold.

Formula:

Region-growing expands from a seed point p by adding neighbors q if:

$$|I(p) - I(q)| < T$$

where T is a predefined intensity threshold.

Implementation in Code:

```
_, thresh = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)
labeled, num_features = ndi_label(thresh)
```

4.2 Connected Component Analysis (CCA)

Description:

Connected Component Analysis (CCA) identifies **contiguous regions** in a binary image by assigning unique labels to connected pixel groups. This helps filter out small, noisy components and extract significant objects.

Formula:

CCA assigns labels to connected components using:

$$L(p) = \min\left(L(N(p))\right), \quad \text{if } I(p) = 1$$

where $N(p)$ is the neighborhood of pixel p.

4.3 Results & Impact

- **Region-growing** improved object completeness by bridging gaps in under-segmented areas.
- **CCA** effectively removed spurious noise, enhancing object separation.
- The final segmented objects are more **refined, clear, and usable for further processing** like bounding box extraction or classification.

5. Results and Discussion

5.1 Results and Discussion:

To evaluate the segmentation performance, we employed **quantitative metrics** such as **Intersection over Union (IoU), Dice coefficient, and pixel accuracy**. These metrics provide an **objective assessment** of the effectiveness of each segmentation technique.

Pixel Accuracy

Pixel accuracy measures the proportion of correctly classified pixels in the segmented image compared to the ground truth. A higher accuracy indicates better segmentation performance.

Formula:

$$IoU = \frac{|A_{GT} \cap A_P|}{|A_{GT} \cup A_P|}$$

This metric reflects the overall correctness of pixel classification in the segmentation pipeline.

Pixel Accuracy

IoU evaluates the **overlap** between the segmented regions and the ground truth. It is computed as follows:

Formula:

$$PA = \frac{TP + TN}{TP + TN + FP + FN}$$

A higher IoU value signifies a greater agreement between the predicted segmentation and the actual objects in the image.

Dice Coefficient:

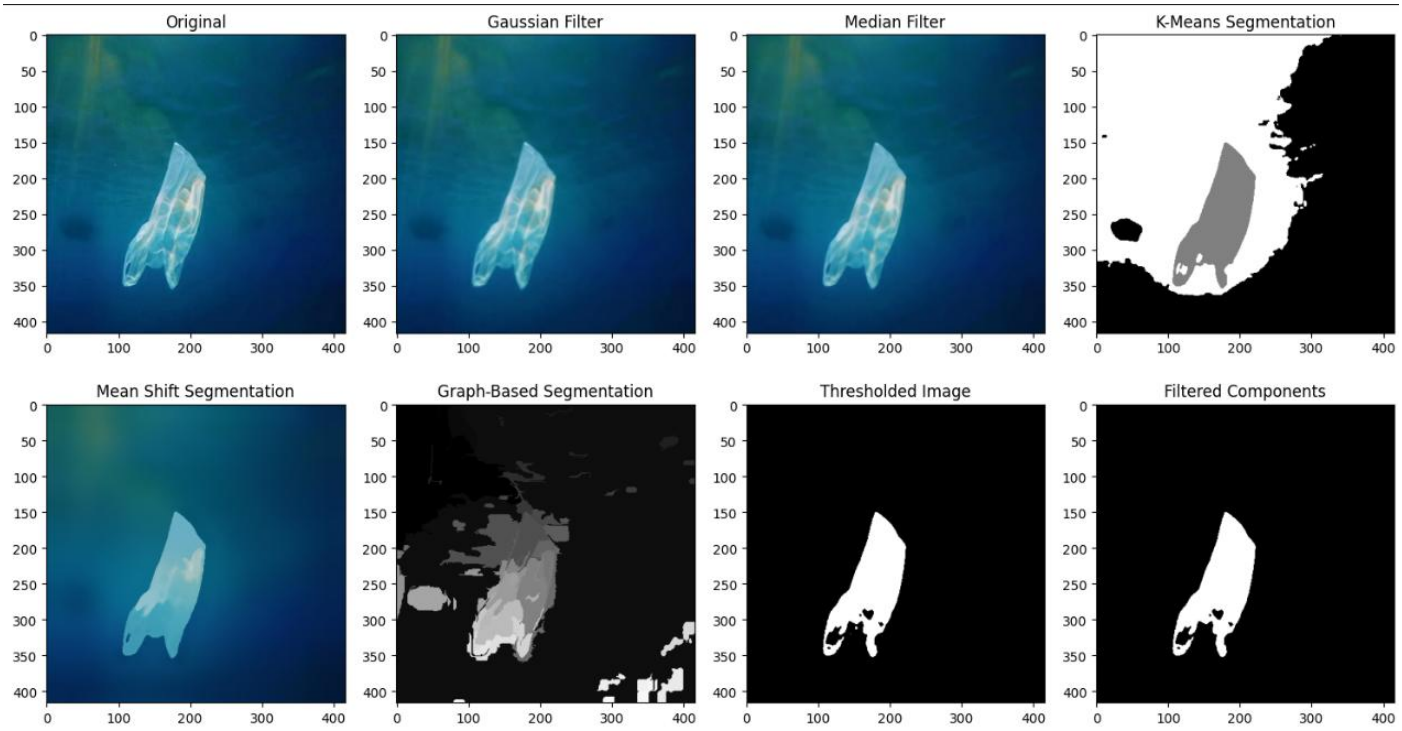
The Dice coefficient is another similarity metric that assesses segmentation performance, particularly balancing **precision and recall**:

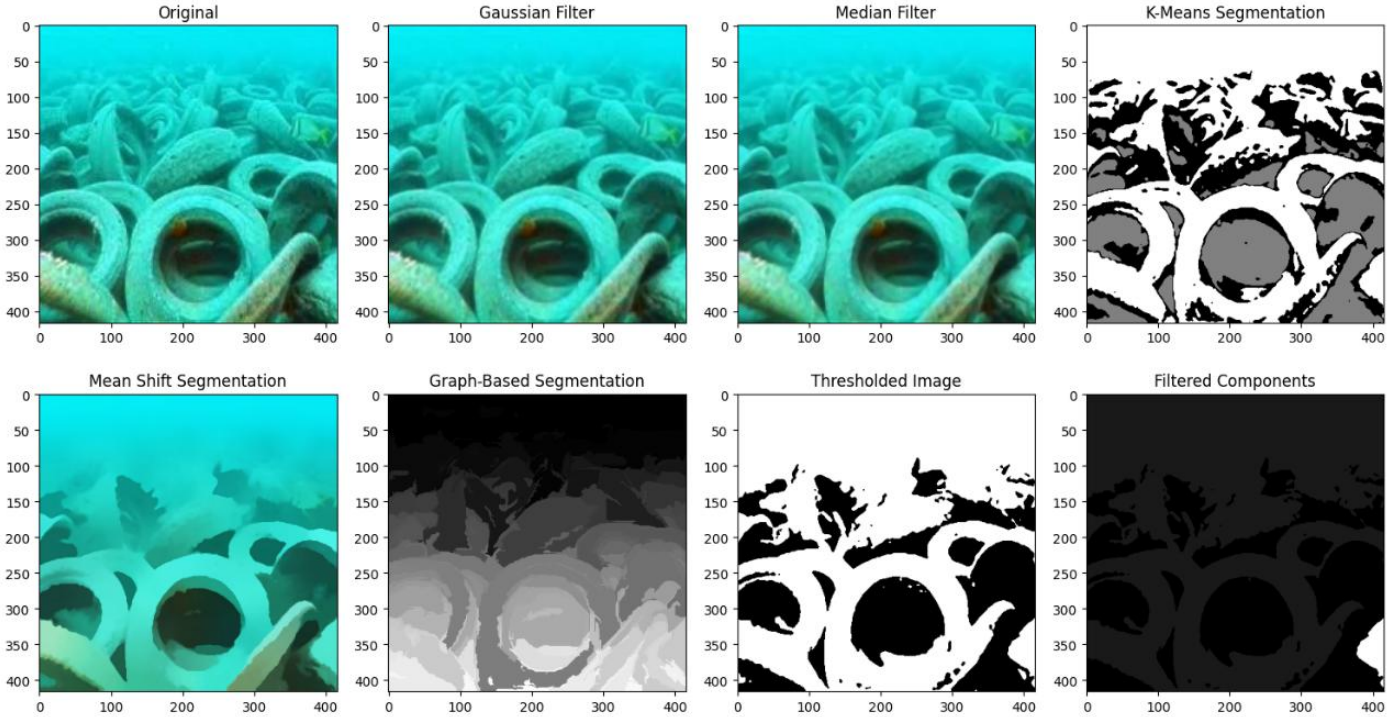
Formula:

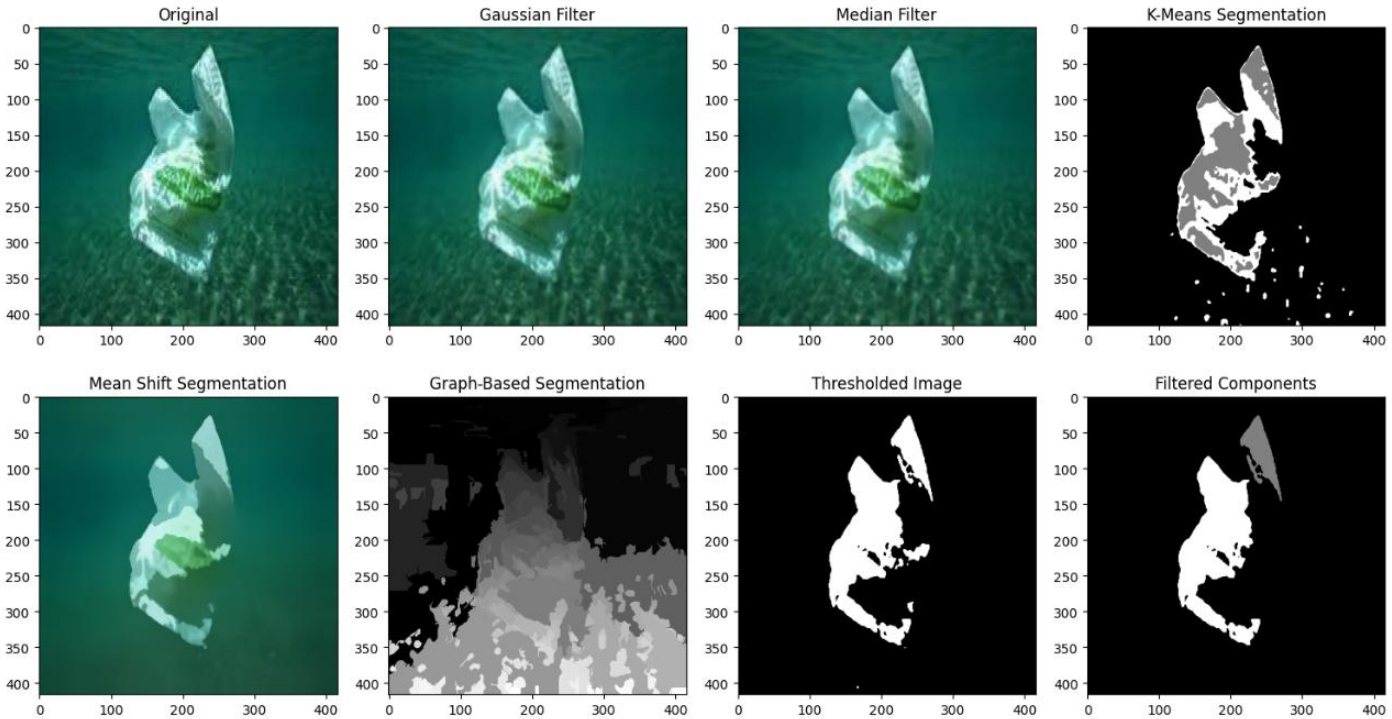
$$Dice = \frac{2|A_{GT} \cap A_P|}{|A_{GT}| + |A_P|}$$

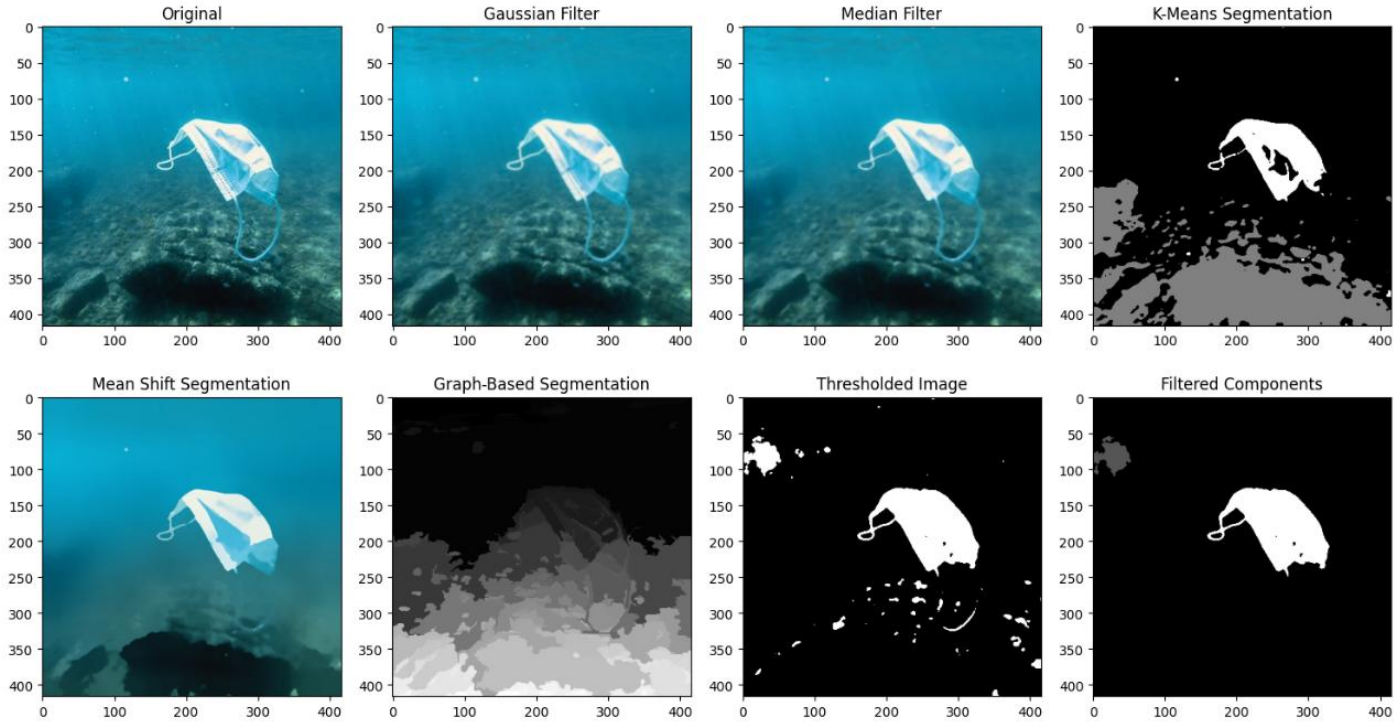
A value closer to **1** suggests a high degree of overlap, indicating **excellent segmentation performance**.

5.2 Visual Comparisons:




Original
Gaussian Filter
Median Filter
K-Means Segmentation
Mean Shift Segmentation
Graph-Based Segmentation
Thresholded Image
Filtered Components


Original
Gaussian Filter
Median Filter
K-Means Segmentation
Mean Shift Segmentation
Graph-Based Segmentation
Thresholded Image
Filtered Components



5.3 Tabular Comparisons:

Image	Filter	IoU
uwg_g-244__fliph_jpg.rf.d5524e6d1ef5df6a12d26e71fcc616b3.jpg	Gaussian	0.373699843
uwg_g-244__fliph_jpg.rf.d5524e6d1ef5df6a12d26e71fcc616b3.jpg	Median	0.373699843
uwg_g-244__fliph_jpg.rf.d5524e6d1ef5df6a12d26e71fcc616b3.jpg	K-Means	0.066989876
uwg_g-244__fliph_jpg.rf.d5524e6d1ef5df6a12d26e71fcc616b3.jpg	Mean Shift	0.373699843
uwg_g-244__fliph_jpg.rf.d5524e6d1ef5df6a12d26e71fcc616b3.jpg	Graph-Based	0.386490088
uwg_g-244__fliph_jpg.rf.d5524e6d1ef5df6a12d26e71fcc616b3.jpg	Region Processing	0.954245334
uwg_g-1186_jpeg_jpg.rf.cafb09ea4242b3f00e853eb10a72f25a.jpg	Gaussian	0.583198502
uwg_g-1186_jpeg_jpg.rf.cafb09ea4242b3f00e853eb10a72f25a.jpg	Median	0.583198502
uwg_g-1186_jpeg_jpg.rf.cafb09ea4242b3f00e853eb10a72f25a.jpg	K-Means	0.362246903
uwg_g-1186_jpeg_jpg.rf.cafb09ea4242b3f00e853eb10a72f25a.jpg	Mean Shift	0.583198502
uwg_g-1186_jpeg_jpg.rf.cafb09ea4242b3f00e853eb10a72f25a.jpg	Graph-Based	0.579823872
uwg_g-1186_jpeg_jpg.rf.cafb09ea4242b3f00e853eb10a72f25a.jpg	Region Processing	0.983007352
uwg_g-15__flipv_jpg.rf.139a0ed23ac4e0662b22ddd7034601ba.jpg	Gaussian	0.229994915
uwg_g-15__flipv_jpg.rf.139a0ed23ac4e0662b22ddd7034601ba.jpg	Median	0.229994915
uwg_g-15__flipv_jpg.rf.139a0ed23ac4e0662b22ddd7034601ba.jpg	K-Means	0.379960478
uwg_g-15__flipv_jpg.rf.139a0ed23ac4e0662b22ddd7034601ba.jpg	Mean Shift	0.229994915
uwg_g-15__flipv_jpg.rf.139a0ed23ac4e0662b22ddd7034601ba.jpg	Graph-Based	0.23474112
uwg_g-15__flipv_jpg.rf.139a0ed23ac4e0662b22ddd7034601ba.jpg	Region Processing	0.95862017
uwg_g-293__flipv_jpg.rf.6d5f6e16b6f8eb1b10a52b699b132b0a.jpg	Gaussian	0.552901951
uwg_g-293__flipv_jpg.rf.6d5f6e16b6f8eb1b10a52b699b132b0a.jpg	Median	0.552901951
uwg_g-293__flipv_jpg.rf.6d5f6e16b6f8eb1b10a52b699b132b0a.jpg	K-Means	0.367986216
uwg_g-293__flipv_jpg.rf.6d5f6e16b6f8eb1b10a52b699b132b0a.jpg	Mean Shift	0.552901951
uwg_g-293__flipv_jpg.rf.6d5f6e16b6f8eb1b10a52b699b132b0a.jpg	Graph-Based	0.555705267
uwg_g-293__flipv_jpg.rf.6d5f6e16b6f8eb1b10a52b699b132b0a.jpg	Region Processing	0.981522318

5.4 Method Comparisons:

Segmentation Method	Observations
K-Means Segmentation	Produced coarse segmentation with clear but rigid boundaries. Struggled with complex object shapes and exhibited inconsistencies in segmenting overlapping regions.
Mean-Shift Segmentation	Generated smoother and more natural object boundaries but occasionally blurred finer details.
Graph-Based Segmentation	Provided the most detailed segmentation with well-defined boundaries, making it the best choice for object extraction.

6. Our Approach

Our approach to underwater plastic pollution image segmentation follows a structured and methodical pipeline designed to enhance image quality, apply diverse segmentation techniques, and evaluate their effectiveness. The process begins with selecting 10 random images from the dataset, which are then converted to RGB format for consistency and compatibility with various image processing methods. Given the challenging nature of underwater imagery, which often includes noise, low contrast, and varying lighting conditions, we applied noise reduction techniques to improve image clarity. Gaussian filtering was used to smooth the image while preserving edges, whereas median filtering effectively removed salt-and-pepper noise, making subsequent segmentation more robust.

To segment the plastic waste regions, we experimented with multiple techniques to capture different aspects of the image structure. K-Means clustering was applied to categorize pixels into distinct groups based on intensity values, helping to differentiate plastic waste from the background. Mean Shift filtering, a non-parametric clustering technique, was used to smooth the image while preserving edges, allowing for more coherent region formation. Graph-Based segmentation, specifically the Felzenszwalb algorithm, was employed to partition the image based on local contrast, ensuring that plastic waste regions were distinctly separated from surrounding textures. Additionally, we applied thresholding techniques to create binary masks of potential plastic waste regions, followed by connected component analysis (CCA) to label and extract individual objects. Small, noisy regions were further removed using region-based processing to refine segmentation accuracy.

To objectively assess the effectiveness of these segmentation techniques, we computed three key evaluation metrics: Intersection over Union (IoU) to quantify the overlap between the predicted segmentation and ground truth, the Dice Coefficient to measure similarity, and Pixel Accuracy to evaluate the proportion of correctly classified pixels. The results were systematically stored in a pandas DataFrame and saved as a CSV file for further analysis. To facilitate visual comparisons, we saved processed images at each stage and displayed them in a grid format, showcasing the original image alongside the various segmentation outputs. This structured approach ensures a comprehensive evaluation of different segmentation techniques, allowing for informed selection of the most effective method for detecting plastic waste in underwater environments.

7. Conclusion and Future Work

Conclusion

Our approach to underwater plastic waste segmentation integrates noise reduction, multiple segmentation techniques, and evaluation metrics to systematically identify plastic regions in underwater images. Through the application of Gaussian and median filtering, we enhanced image quality, reducing the impact of noise while preserving crucial details. We then experimented with diverse segmentation methods, including K-Means clustering, Mean Shift filtering, Felzenszwalb's graph-based segmentation, and thresholding with connected component analysis. The performance of each method was assessed using IoU, Dice Coefficient, and Pixel Accuracy, providing quantitative insights into their effectiveness. Our results indicate that no single method is universally optimal, as each has strengths and limitations based on image complexity, plastic object texture, and background variations. However, region-based segmentation combined with noise filtering demonstrated promising results in separating plastic waste from underwater backgrounds. The final processed images and evaluation metrics offer a structured foundation for further optimization in underwater plastic detection tasks.

Future Work

While our segmentation framework effectively identifies plastic waste in underwater images, there are several areas for improvement and further exploration. First, integrating deep learning-based segmentation models such as U-Net, Mask R-CNN, or DeepLabV3+ can enhance accuracy by leveraging learned feature representations. Training these models on a large annotated dataset could improve the generalization of plastic waste detection across varying underwater conditions. Additionally, incorporating hyperspectral imaging

techniques or domain adaptation methods can help address challenges like color distortions, water turbidity, and varying lighting conditions. Another promising direction is the fusion of multimodal data, such as sonar imaging or spectral reflectance analysis, to improve segmentation robustness. Further, real-time plastic waste detection using optimized models deployed on edge devices or autonomous underwater vehicles (AUVs) can enable practical environmental monitoring and cleanup efforts. Finally, refining evaluation strategies by including human expert validation or integrating weakly supervised learning could enhance segmentation performance and applicability in real-world scenarios.